UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **STIC Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Nabil FAKHFAKH

Thèse dirigée par **Patrice MOREAUX** et **Hervé VERJUS**
Co-encadrée par **Frédéric POURRAZ**

préparée au sein du **LISTIC**
dans **l'École Doctorale SISEO**

Une approche orientée utilisateur pour la supervision des orchestrations de services

Thèse soutenue publiquement le **6 juin 2012**,
devant le jury composé de :

Mme Laurence DUCHIEN
Professeur des Universités à l'Université de Lille 1, Rapporteur
Mme Lynda MOKDAD
Professeur des Universités à l'Université de Paris-Est Créteil, Rapporteur
Mme Valérie BOTTA-GENOULAZ
Professeur des Universités à l'INSA de Lyon, Président
Mme Irène ABI-ZEID
Professeure agrégée à l'Université Laval (Canada), Examinateur
M. Vincent CLIVILLÉ
Maître de Conférences à l'Université de Savoie, Examinateur
M. Patrice MOREAUX
Professeur des Universités à l'Université de Savoie, Directeur de thèse
M. Hervé VERJUS
Maître de Conférences à l'Université de Savoie, Co-directeur de thèse
M. Frédéric POURRAZ
Maître de Conférences à l'Université de Savoie, Co-encadrant

Université Joseph Fourier / Université Pierre Mendès France /
Université Stendhal / Université de Savoie / Grenoble INP

À mes parents ...

Remerciements

Je tiens à exprimer mes remerciements à Mme Laurence DUCHIEN, Professeur des universités à l'Université de Lille 1 et à Mme Lynda MOKDAD, Professeur des universités à l'Université Paris-Est Créteil pour m'avoir fait l'honneur de rapporter les travaux de cette thèse.

J'associe à ces remerciements Mme Valérie BOTTA-GENOULAZ, Professeur des universités à l'Institut Nationale des Sciences Appliquées de Lyon, d'avoir présidé mon jury de thèse. Mes remerciements s'adressent également à Mme Irène ABI-ZEID, Professeur agrégée à l'Université Laval au Canada, et M. Vincent CLIVILLÉ, Maître de conférences à l'Université de Savoie, d'avoir examiné mes travaux de thèse. J'adresse particulièrement un grand merci à M. Vincent CLIVILLÉ pour tous ses conseils, les discussions extrêmement enrichissantes que j'ai eues la chance de mener avec lui, sans oublier l'opportunité qu'il m'a accordée de rencontrer M. Jean-Marie DE CORTE, Professeur à l'Université de Mons-Hainaut en Belgique et l'un des créateurs de MACBETH.

Je tiens à exprimer ma profonde gratitude et mes remerciements les plus sincères à M. Patrice MOREAUX, Professeur des universités à l'Université de Savoie, M. Hervé VERJUS, Maître de conférences à l'Université de Savoie et M. Frédéric POURRAZ, Maître de conférences à l'Université de Savoie, pour avoir dirigé mes travaux de thèse mais avant tout pour m'avoir accordé leur confiance et m'avoir attribué le sujet de cette thèse. Les trois années que j'ai passées sous leur encadrement fut une expérience très riche tant au niveau scientifique et professionnel, qu'au niveau personnel. Je tiens à remercier encore mes encadrants de thèse pour la qualité des échanges qui étaient toujours constructifs et pour les débats passionnants que nous avons eus.

Je remercie Mme Aline BERGER, chef de projets à THÉSAME et coordinatrice du projet MES, les dirigeants et tous les collaborateurs des entreprises ALPHA-3i, Quasar solutions, M1i, Courbon, Carl Software, Jam France, Cincom, nos collègues du laboratoire DISP de l'INSA de Lyon, toutes les personnes qui ont participé au projet MES, pour m'avoir permis de réaliser mes travaux dans un contexte professionnel et scientifique riche, varié. Les collaborations que nous avons su mettre en place, nos rencontres périodiques et fructueuses ont été des atouts pour la réussite du projet MES et pour la réalisation de la solution MESTRIA. À titre personnel, l'opportunité de me confronter au secteur de l'édition de progiciels industriels, de participer à certains salons industriels, salons de solutions logicielles m'a beaucoup apporté. Je remercie également M. Brice CARNUS, ingénieur et coordinateur technique durant les deux premières années du projet MES, pour nos différentes collaborations au sein du projet.

Mes remerciements vont également à tous les collègues du laboratoire LISTIC pour les moments agréables partagés durant cette période.

Enfin, je remercie de tout mon cœur mes parents. Grâce à leurs efforts, leurs sacrifices et leur soutien qui ne cessent encore, j'ai pu suivre toutes mes études.

Résumé : La qualité de service est devenue aujourd'hui une notion incontournable dans le développement des applications logicielles, en particulier dans le cadre des architectures orientées services. Les travaux de cette thèse se focalisent sur la supervision de la qualité de service des applications orientées services, définies sous forme d'orchestrations de services. L'approche de supervision proposée dans ce contexte est générique. Elle repose sur des patrons de flux de contrôle des orchestrations de services pouvant être implémentés en intégralité ou en partie par tout langage d'orchestration de services. D'autre part, elle ne pose aucune restriction, ni sur les attributs qualité à surveiller par le système de supervision, ni sur leurs représentations. Cette approche de supervision se distingue des approches existantes par l'exploitation d'un modèle de préférences orienté utilisateur, permettant de représenter fidèlement la satisfaction de ce dernier. Le degré de satisfaction, issu du modèle de préférences, constitue une information de haut niveau représentant la qualité globale de l'orchestration étudiée. Sur la base de ce degré de satisfaction, de nouvelles stratégies de surveillance sont proposées afin de satisfaire les attentes de l'utilisateur. L'élaboration du modèle de préférences exploite la méthode d'aide à la décision multi-critères MACBETH étendue avec l'opérateur d'agrégation de l'intégrale de Choquet 2-additive.

Une illustration de l'approche de supervision a été réalisée sur une orchestration de services, représentant un processus industriel dans le domaine du pilotage d'atelier de production. Les travaux de cette thèse ont été réalisés dans le cadre d'un projet *R&D* regroupant sept éditeurs de progiciels dans le domaine du MES (*Manufacturing Execution System*).

Mots-clefs : supervision, orchestrations de services, degré de satisfaction, qualité de service, règles d'agrégation de patrons de *workflow*, méthode d'aide à la décision multicritère, architectures orientées services.

Abstract: Quality of Service (QoS) is an important issue today in the developement of software applications, especially in the context of Service-Oriented Architectures (SOA).

The work of this thesis focuses on QoS supervision of service-oriented applications, defined as service orchestrations. The proposed supervision approach is generic. It is based on workflow control-flow patterns, which can be entirety or partillay implemented by any service orchestration language. On the other hand, it does not make any restrictions, neither on the monitored QoS attributes , nor on their representations. This supervision approach is based on a user-oriented preferences model, that represents faithfully the user satisfaction. The satisfaction degree derived from the preferences model is a high-level information representing the overall quality of the orchestration. New monitoring strategies are proposed on the basis of this satisfaction degree in order to satisfy user expectations. The elaboration of the preferences model uses the MACBETH method extended to the 2-additive Choquet integral operator as a multi criteria decision aiding method.

An illustration of the approach is carried out on a service orchestration representing a Manufacturing Execution System (MES) process. This work was realized in a *R&D* project involving seven software vendors in the field of MES.

Keywords: Supervision, Service Orchestration, Satisfaction Degree, Quality of Service, Workflow Pattern Aggregation Rules, Multicriteria Decision Aiding Method, Service-Oriented Architectures.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Contexte

L'évolution constante du marché, la compétitivité accrue entre les entreprises et les besoins variables des clients poussent les entreprises à s'adapter en permanence et à être de plus en plus dynamiques et réactives à la diversification des offres commerciales, aux changements technologiques, etc. Cela nécessite souvent de franchir les barrières de l'entreprise et de prendre en compte les activités et les processus métiers des partenaires. Or, les activités et les processus métiers des entreprises sont pilotés par des applications (logicielles). Ceci implique que si certaines (parties des) applications de l'entreprise restent pérennes, d'autres doivent être améliorées, adaptées voire remplacées rapidement pour que l'entreprise soit réactive. L'intégration difficile de ces changements au sein des applications peut être un frein à la réactivité/compétitivité des entreprises. Il faut donc pouvoir garder l'existant tout en permettant le développement et l'amélioration d'autres applications. Autrement dit, il ne faut pas que l'évolution des processus métiers (activité de l'entreprise) soit dépendante des changements des technologies de leur mise en œuvre, mais des modifications des processus métiers eux-mêmes. Ce chalenge constitue un des objectifs primordiaux que tentent d'atteindre les architectures orientées services.

Les architectures orientées services représentent une démarche, un style de conception qui permet aux entreprises de développer, interconnecter et maintenir leurs applications à moindre coût et délais [Papazoglou and Heuvel, 2007]. Les architectures orientées services, comme style de conception, sont indépendantes de toute technologie spécifique. C'est dans ce sens où leur but réside à faire évoluer les processus métiers indépendamment des changements technologiques. Ce style de conception architecturale repose sur la notion de « service ». L'objectif fondamental d'un service, dans une architecture orientée services, est de représenter une unité complète, réutilisable, de fonctionnalité(s) métier(s) [Papazoglou and Heuvel, 2007]. Il est constitué d'une paire : une interface et une implémentation. L'interface d'un service définit son identité et expose les fonctionnalités qu'il offre. Elle est standard et est indépendante de toute plateforme technologique. L'implémentation

d'un service, quant à elle, est la mise en œuvre de la ou les fonctions que le service est chargé d'assurer. Les services sont ainsi autonomes dans le sens où leur fonction est réalisée d'une manière invisible par l'extérieur et que seul leur résultat attendu peut être prévu. Ils peuvent ainsi être améliorés/adaptés d'une façon transparente à leurs utilisateurs. Les applications à base de services (reposant sur une architecture orientée services) bénéficient donc de l'avantage d'être maintenues et adaptées sans que ces changements impactent l'existant. En outre, de nouveaux services peuvent être intégrés/composés pour répondre à des nouveaux besoins et étendre ainsi les processus métiers de l'entreprise. On parle ainsi de composition de services et, dans notre étude, d'*orchestration de services*.

En disposant d'un ensemble (une cartographie) de services métiers, les entreprises peuvent orchestrer ces services à leur convenance pour répondre aux offres spécifiques et variables des clients et du marché. Les entreprises peuvent également *faire évoluer* leurs processus métiers en intégrant des services externes ou en créant des alliances avec d'autres entreprises pour répondre, ensemble, aux offres du marché les plus complexes [Fakhfakh et al., 2010]. C'est typiquement le contexte du projet *Manufacturing Execution System, MES* [Khamès, 2010] dans lequel s'inscrivent en partie les travaux de cette thèse.

Plusieurs travaux dans la littérature [Cîmpan and Verjus, 2005, Pourraz and Verjus, 2007, Ben Halima et al., 2008, Qiao et al., 2009, Cîmpan et al., 2009] ont abordé cette notion d'évolution. Outre l'évolution des orchestrations de services qui est requise pour étendre les processus métiers et répondre ainsi à des nouveaux besoins, un autre aspect aussi important consiste à réaliser l'évolution des orchestrations de services dans un but de remédier aux aléas qui ont lieu, en particulier au cours de l'exécution [Ben Halima et al., 2008, Qiao et al., 2009]. Ces aléas sont relativement plus fréquents dans les architectures orientées services du fait de la nature répartie des services et de leur caractère autonome. En effet, les services sont des entités autonomes, potentiellement déployées et exécutées sur des plateformes externes (à l'utilisateur) et sont donc non maîtrisables par l'utilisateur. Par conséquent, lorsqu'un aléa survient (au cours de l'exécution), l'utilisateur n'est pas forcément averti afin de prendre les décisions nécessaires et donc le bon déroulement de son processus métier peut être mis en jeu. Ceci implique que pour faire évoluer les orchestrations de services suite à la présence d'un aléa, il faut d'abord le détecter ou mieux encore anticiper sa potentielle apparition afin d'éviter les conséquences qu'il pourrait engendrer. C'est essentiellement la responsabilité d'un système de supervision qui consiste à surveiller l'exécution des orchestrations de services (réalisant les processus métiers) et de prévenir et/ou à signaler les anomalies qui peuvent avoir lieu au cours de l'exécution. C'est dans ce contexte que se situent les travaux de cette thèse. En particulier, nous nous intéressons à la supervision des anomalies liées à la qualité des services et à la qualité des orchestrations de services en général.

1.2 Problématique

La notion de qualité est devenue aujourd'hui incontournable dans le développement des applications à base de services. Elle représente un critère important aussi bien pour les utilisateurs (clients, entreprises, etc) que pour les fournisseurs de services, au point que la notion de contrats portant sur la qualité est devenue indispensable. Ainsi, les systèmes de supervision de la qualité prennent de l'importance et se voient à leur tour indispensables lorsque l'on adopte une architecture orientée services. Nous nous focalisons dans nos travaux sur la supervision de la qualité des orchestrations de services. De ce fait, il convient d'abord d'évaluer la qualité des orchestrations de services pour pouvoir la surveiller.

Évaluation des attributs qualité des orchestrations de services au cours de l'exécution

La qualité des orchestrations est représentée par des valeurs d'attributs qualité comme par exemple le temps de réponse, le coût, la disponibilité, la fiabilité, etc. Plusieurs travaux [Zhong and Qi, 2006, Gallotti et al., 2008, Sato and Trivedi, 2007, Cardoso et al., 2002, C. Jaeger, 2007, Hwang et al., 2007, Rosenberg et al., 2009] ont traité ce point et ont proposé des approches permettant de déduire la valeur de chaque attribut qualité correspondant à l'orchestration globale, à partir des valeurs des attributs qualité relatives aux services impliqués dans l'orchestration. Certains de ces travaux [Zhong and Qi, 2006, Gallotti et al., 2008, Sato and Trivedi, 2007] sont restreints à certains attributs qualité, notamment au temps de réponse et/ou à la fiabilité, d'autres [Cardoso et al., 2002, C. Jaeger, 2007, Hwang et al., 2007, Rosenberg et al., 2009] se limitent à quelques structures de modélisation de la logique de contrôle des orchestrations de services. De surcroît, la limitation majeure, à notre sens, de la plupart des travaux récents, réside dans la représentation des attributs qualité, notamment pendant la conception et au cours de l'exécution (concernant la partie non exécutée de l'orchestration). À titre d'exemple, le temps de réponse est souvent représenté par une moyenne statistique, ce qui nous semble une estimation non précise, notamment dans le contexte que nous avons présenté précédemment où les valeurs des attributs qualité peuvent dévier de leur valeurs moyennes à l'exécution. Ceci bien entendu aura des répercussions, au niveau du système de supervision, sur la gestion de la qualité des orchestrations qui risque de ne pas être conforme aux attentes de l'utilisateur (client, entreprise, etc).

D'autre part, la surveillance de la qualité de l'orchestration nécessite l'évaluation de sa qualité tout au long de l'exécution. Tous les travaux précédemment cités ciblent la phase de conception de l'orchestration (pour des fins de sélection et de composition des services). Bien que certains travaux [Canfora et al., 2008, Szydlo and Zielinski, 2008, Qiao et al., 2009] aient adressé cette problématique, leurs approches présentent certaines limitations, notamment dans la représentation des estimations des valeurs des attributs qualité, souvent prises comme des moyennes statistiques. Là encore, cela peut provoquer des erreurs

d'estimation non négligeables, susceptibles d'engendrer des interventions (réalisées par le système de supervision) inutiles ou au contraire insuffisantes pour améliorer la qualité de l'orchestration.

Interprétation de la qualité globale des orchestrations de services

Les travaux cités précédemment, qui permettent l'évaluation des attributs qualité des orchestrations, présentent des limitations quand il s'agit d'interpréter globalement les valeurs de ces différents attributs qualité, notamment en présence de multiples attributs qualité. Autrement dit, étant donné les valeurs de chacun des attributs qualité qui peuvent être éventuellement un mélange de bonnes et de mauvaises valeurs, que peut-on dire de la qualité globale de l'orchestration ? À quel point est-elle satisfaisante ou inversement insatisfaisante par rapport aux contraintes de qualité spécifiées ? La réponse à cette question peut être relativement complexe en raison de la multiplicité des attributs qualité et des limites des capacités cognitives de l'utilisateur [Bouyssou et al., 2006].

D'autre part, la notion de dégradation de qualité, qui est la principale tâche à identifier par le système de supervision, est relative au contexte d'utilisation de l'orchestration et aux exigences de l'utilisateur en termes de qualité. En effet, d'un processus à un autre, l'importance d'un même attribut qualité du point de vue utilisateur peut varier. Par exemple, pour un processus de traitement de grandes quantités d'informations, le temps de réponse est moins important que l'exactitude des résultats (fiabilité). Par conséquent, une dégradation du temps de réponse peut être tolérée contrairement à une dégradation de la fiabilité tout en préservant la satisfaction du client en termes de qualité perçue. Il est donc important d'associer des poids d'importance à chacun des attributs qualité pour représenter les préférences de l'utilisateur et par la suite pour une meilleure interprétation. Certains travaux [Canfora et al., 2008, Szydlo and Zielinski, 2008] ont proposé, dans un contexte différent (qui est la sélection des services), une solution qui répond partiellement à cette problématique. Elle consiste à définir une fonction objectif représentée par une moyenne pondérée par les poids d'importance associés aux attributs qualité. Cette fonction objectif a pour vocation d'agréger les valeurs des différents attributs qualité (normalisées) permettant ainsi d'avoir une seule information globale sur la qualité de l'orchestration. Cependant, ces travaux présentent certaines limitations. Outre les modèles d'orchestrations considérés dans ces travaux qui se limitent à certains cas particuliers de modèles d'orchestration (par exemple dans [Szydlo and Zielinski, 2008], seulement les services en séquence sont pris en compte), le résultat de leur approche d'agrégation ne nous semble pas assez satisfaisant [Fakhfakh et al., 2012]. D'une part, la plupart des travaux exploitant une fonction objectif pour l'agrégation de la qualité (que ce soit la qualité de l'orchestration [Canfora et al., 2008] ou la qualité des services [Taher et al., 2005b, Herssens et al., 2008]) utilisent des fonctions d'utilité linéaires pour la normalisation des valeurs des attributs qualité. Ceci suppose que la satisfaction des utilisateurs est linéaire vis-à-vis des valeurs des attributs qualité, ce qui n'est pas forcément le cas [Fakhfakh et al., 2011d, Fakhfakh et al., 2011e] et

peut engendrer des conséquences sur le traitement réalisé ultérieurement. D'autre part, la fonction objectif dans ces travaux est souvent basée sur la moyenne pondérée, un opérateur d'agrégation qui ne permet pas de représenter certaines préférences (de l'utilisateur) relatives aux attributs qualité. En particulier, cet opérateur d'agrégation se montre incapable de représenter des dépendances entre les attributs qualité [Fakhfakh et al., 2011b, Grabisch and Labreuche, 2008]. Cela implique, encore une fois, un manque de fidélité dans la représentation des préférences de l'utilisateur. Cette infidélité résultant de la méthode de normalisation ou de la fonction objectif elle-même, va bien entendu être répercutée sur les traitements réalisés par la suite, notamment dans notre cas, sur les décisions prises par le système de supervision, chargé de surveiller la qualité de l'orchestration. Cela pourrait être à l'origine d'actions stériles réalisées par le système de supervision, ce qui va à l'encontre des attentes des utilisateurs.

Surveillance de la qualité globale des orchestrations de services

Dans les travaux actuels [Canfora et al., 2008, Szydlo and Zielinski, 2008, Ben Halima et al., 2008, Qiao et al., 2009] relatifs à la gestion de la qualité des orchestrations, le système de supervision surveille la déviation des valeurs (perçues) des attributs qualité au regard des exigences définies (contraintes fixées sur les valeurs des attributs qualité). Aucun des travaux actuels n'inclut les préférences des clients dans la surveillance de la qualité des orchestrations de services. La norme ISO 9001:2008 [ISO/TC 176, 2008] précise pourtant qu'un des indicateurs de performance d'un système de gestion de la qualité est la surveillance des informations relatives à la perception du client sur *le niveau de satisfaction de ses exigences.* Les informations dégagées grâce à la surveillance et à la mesure de la satisfaction du client peuvent aider les organisations à prendre des décisions visant à maintenir ou à accroître la satisfaction des clients et à atteindre leurs objectifs (avoir la confiance des clients, attirer plus de clients, générer des bénéfices commerciaux, etc). Ainsi, pour un contrôle conforme aux attentes du client, le système de supervision doit prendre en considération les *préférences* de l'utilisateur sur les attributs qualité. Cette problématique n'a pas encore été étudiée à notre connaissance.

1.3 Contributions

Face aux problématiques exposées, les contributions des travaux de cette thèse s'inscrivent dans le cadre d'une approche de supervision de la qualité globale des orchestrations de services en cours d'exécution (voir figure 1.1). Les apports de nos travaux relatifs à chacune des problématiques identifiées précédemment comportent quatre volets.

Le premier volet cible la première problématique présentée. Dans ce cadre, nous proposons une approche d'agrégation de chacun des attributs qualité durant les différentes phases du cycle de vie de l'orchestration, à savoir pendant la phase de conception, au cours de l'exécution et à la fin de l'exécution. Nous étudions également les travaux existants et

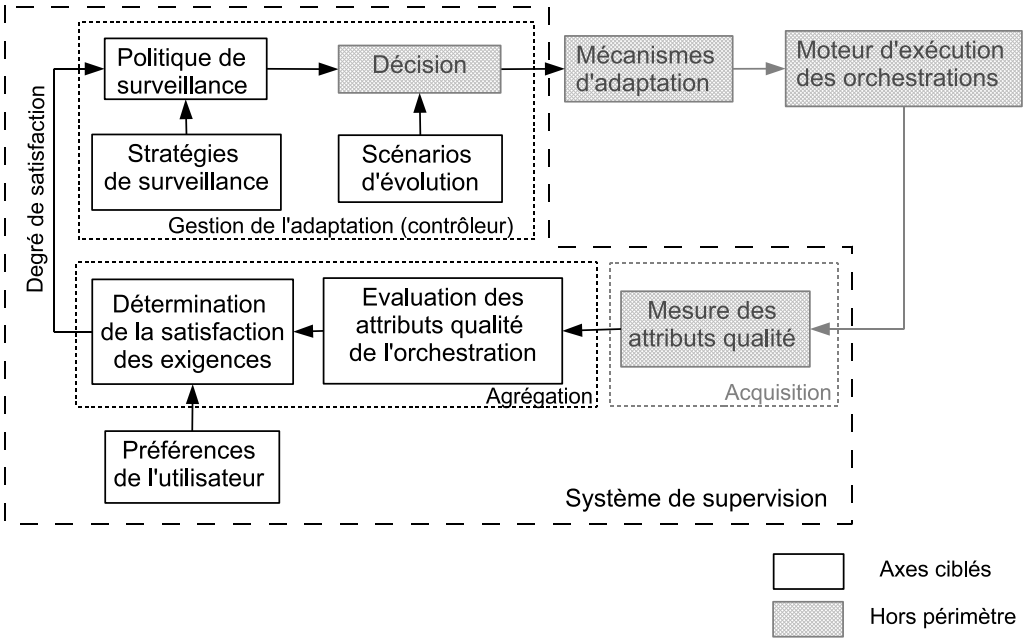


Figure 1.1 – Principales contributions de la thèse

nous proposons de nouvelles règles d'agrégation qui sont aujourd'hui manquantes pour certains attributs qualité.

Dans le but d'améliorer les travaux issus de la littérature, nous construisons *un modèle de préférences* qui permet de refléter au mieux la satisfaction réelle de l'utilisateur (consommateur des services, décideur, etc). Ce modèle de préférences est utilisé par la suite dans le calcul de la valeur agrégée (utilité globale), que nous appelons *degré de satisfaction*, représentant la qualité globale de l'orchestration.

Afin que le système de supervision soit fidèle aux attentes de l'utilisateur, nous proposons des stratégies de surveillance ainsi que de nouvelles politiques de surveillance de la qualité de l'orchestration, basées sur le degré de satisfaction. Nous présentons dans ce contexte quelques scénarios d'évolution qui ont vocation à améliorer la qualité des orchestrations de services.

En dernier lieu, nous présentons la démarche de mise en œuvre des architectures orientées services dans un contexte réel qu'est celui du projet MES.

1.4 Organisation du document

Ce mémoire est structuré de la manière suivante.

Dans le chapitre 2, nous présentons une introduction aux architectures orientées services et aux orchestrations de services. Nous abordons ensuite la qualité de service dans

les architectures orientées services et nous en recensons en particulier les attributs qualité. Nous détaillons également un état de l'art sur les travaux portant sur la qualité des orchestrations de services ainsi que ceux liés à la gestion de la qualité des orchestrations. Nous faisons enfin une synthèse de cet état de l'art au regard de notre problématique.

Nous introduisons, dans le chapitre 3, notre approche globale de supervision et nous fixons les objectifs et le périmètre de nos travaux. Nous proposons ensuite une approche d'agrégation des attributs qualité pour chacune des phases du cycle de vie des orchestrations de services, à savoir pendant la phase de conception, durant l'exécution et à la fin de l'exécution.

Nous présentons par ailleurs dans le chapitre 4, la méthode que nous exploitons pour la mesure du degré de satisfaction des orchestrations au regard des exigences des clients. La méthode comprend également la construction du modèle de préférences relatif à l'utilisateur. Nous exposons ensuite le problème de dépendances (préférentielles) qui peuvent exister et nous détaillons la solution permettant de les prendre en compte. Enfin, nous décrivons notre approche de gestion de la qualité des orchestrations de services en cours d'exécution, incluant les stratégies de surveillance, leur politique de surveillance ainsi que certains scénarios d'évolution permettant l'amélioration de la qualité de l'orchestration.

Nous consacrons le chapitre 5 à la validation de nos travaux. Ainsi, nous présentons une mise en œuvre d'un style de conception architecturale, celui des architectures orientées services, dans un contexte particulier, le projet MES dans lequel s'inscrivent partiellement les travaux de cette thèse. Nous montrons ensuite l'application de notre approche de supervision sur un exemple d'orchestration de services issu du domaine de MES, et nous comparons nos travaux par rapport aux travaux issus de la littérature.

En dernier lieu, nous établissons dans le chapitre 6, un bilan de notre approche de supervision par rapport aux travaux existants et discutons de ses limites. Nous présentons ensuite quelques perspectives qui pourront être envisagées à l'issue de ces travaux de thèse.

Chapitre 2

État de l'art

2.1 Introduction

Aujourd'hui, les technologies de l'information évoluent d'une manière considérable et les demandes des clients sont devenues de plus en plus pointues et diversifiées. Face à ces circonstances, les systèmes logiciels ou à forte composante logicielle ont besoin d'être agiles et capable d'intégrer et de réutiliser dans de fortes contraintes de délai, d'autres fonctionnalités fournies par d'autres systèmes logiciels. Ces besoins de réutilisation et d'agilité sont entre autres les aspects que favorisent principalement les architectures orientées services. Ils sont assurés à travers les notions de « service » et de « composition de services » sur lesquelles se basent les architectures orientées services (section2.2). Une autre notion importante qui a fait son émergence dans les systèmes logiciels basés sur une architecture orientée service, est la qualité de service (section 2.3). Cet aspect de qualité est devenu aujourd'hui incontournable, notamment face à la prolifération des services et à l'exigence accrue des clients. De surcroît, les systèmes logiciels adoptant une architecture orientée service, se basent sur un ensemble de services qui sont potentiellement fournis par différentes organisations. Par conséquent, ils peuvent être distribués à large échelle et opérer sur des plateformes hétérogènes [Sathya et al., 2011, He et al., 2008, Zeng et al., 2003]. Dans ces conditions, la qualité de ces systèmes logiciels (à base de services) dépend de la qualité des services qu'ils intègrent [Hwang et al., 2007, C. Jaeger, 2007, Rosenberg, 2009, Menascé et al., 2010]. Or ces services ne sont pas maîtrisés par l'entité qui les intègre (consommateurs) et peuvent subir des changement de leur niveau de qualité sans avertir les consommateurs [Zheng and Lyu, 2010, Hwang et al., 2007, Taher et al., 2005a]. Dans ce contexte, la surveillance et la garantie de la qualité de ces systèmes logiciels sont devenues un enjeu crucial qui a suscité l'intérêt de plusieurs travaux de recherche [Canfora et al., 2008, Qiao et al., 2009, Ben Halima et al., 2008, Szydlo and Zielinski, 2008] (section 2.4). C'est essentiellement autour de ces thématiques que nous orientons l'état de l'art de nos travaux de recherche.

2.2 Architectures orientées services

2.2.1 Définition et principe

Une architecture orientée service (AOS, en anglais *Service Oriented Architecture*) fait référence à un style de conception de systèmes distribués qui exposent leurs fonctionnalités en services en se focalisant davantage sur la notion du faible couplage entre les services [Srinivasan and Treadwell, 2005]. On trouve plusieurs définitions dans la littérature parmi lesquelles nous retenons les suivantes :

Définition 2.1. Une architecture orientée service est une approche pour construire des systèmes distribués en intégrant des composants qui sont indépendants de la plateforme technologique, des langages d'implémentation et des systèmes d'exploitation. Une AOS délivre les fonctionnalités d'une application sous forme de services qui seront exploités directement par l'utilisateur final ou utilisés pour former d'autres services [Quynh and Thang, 2009].

Définition 2.2. Une architecture orientée service instaure un modèle architectural qui a pour objectif d'améliorer l'efficacité, l'agilité et la productivité d'une entreprise en considérant les services comme le principal moyen à travers lequel la logique d'une solution est représentée à l'appui de la réalisation des objectifs stratégiques [Thomas, 2007, chap.3].

Ces définitions reflètent deux notions importantes des AOS qui sont « *les services* » et « *la composition de services* ». Les services constituent la brique de base sur laquelle reposent les AOS. Un service est une entité autonome qui encapsule une ou plusieurs fonctionnalités appelées *opérations de service*. Ces opérations de service sont indépendantes du contexte ou de l'état d'autres services [Papazoglou and Heuvel, 2007]. Plus précisément, les services sont indépendants de toute plateforme technologique ou langages d'implémentation. Ils exposent leurs fonctionnalités à travers une interface standard (ex. *Web Service Description Language–WSDL* pour les services web) et communiquent via des échanges de messages [Pourraz, 2007]. La figure 2.1 décrit l'interaction entre les principaux acteurs dans les AOS à savoir les clients [1], les fournisseurs de services, et l'annuaire de services. Le fournisseur de service commence par la publication de l'interface du service dans un annuaire de services connu (interaction (1) de la figure 2.1), comme par exemple l'annuaire *Universal Description, Discovery and Integration– UDDI* pour les services web. Cette interface contient les fonctionnalités (opérations de service) offertes par le service et la manière dont elles peuvent être invoquées. En particulier, l'interface décrit les paramètres d'entrée requis pour le fonctionnement de chaque opération de service ainsi que les paramètres de sortie que produit l'opération de service suite à son exécution. Ensuite, un consommateur potentiel peut découvrir dans cet annuaire les services répondant à ses besoins métiers (interaction (2) de la figure 2.1). L'annuaire retourne au consommateur

1. Dans la suite du manuscrit, nous utilisons indifféremment les termes utilisateur et client pour désigner le consommateur des services ou des orchestrations de services, l'utilisateur final, etc.

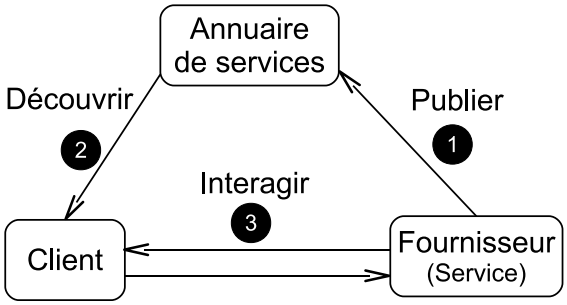


Figure 2.1 – Interaction entre le client, le fournisseur et l'annuaire des services

une liste de services disponibles ainsi que leurs interfaces. Enfin, le consommateur choisit dans la liste de services, le service le plus adéquat et établit une connexion avec le fournisseur de service afin d'invoquer les opérations de service souhaitées (interaction (3) de la figure 2.1). Notons que c'est bien les opérations de service qui sont invoquées et non pas les services. La notion d'invocation de service est un abus de langage.

En plus d'être découverts et invoqués, les services peuvent être composés pour développer des applications logicielles ou d'autres services plus riches en termes de fonctionnalités métiers. La composition de services englobe deux concepts différents : *la chorégraphie* et *l'orchestration*. La chorégraphie permet de décrire la collaboration entre plusieurs services en termes de messages échangés et de règles d'interaction entre les participants d'un processus. La chorégraphie décrit, d'un point de vue global, le comportement observable des participants au sein d'un processus métier [Papazoglou et al., 2006, Peltz, 2003]. L'orchestration, quant à elle, fait référence aux processus exécutables ; contrairement à la chorégraphie. Nos travaux de recherche ciblent particulièrement les processus exécutables, c'est pour cela que la suite du chapitre focalise essentiellement sur la notion d'orchestrations de services.

2.2.2 Orchestrations de services

Les services qui répondent aux besoins métiers ayant été choisis, l'orchestration consiste à organiser ces services afin d'accomplir le processus métier souhaité. Là encore, on trouve de nombreuses définitions des orchestrations de services :

Définition 2.3. L'orchestration fait référence à un processus métier exécutable qui peut interagir aussi bien avec des services web internes qu'avec des services web externes. L'orchestration décrit la manière dont les services web peuvent interagir au niveau message, incluant la logique métier et l'ordre d'exécution des interactions. Ces interactions peuvent s'étendre à plusieurs applications et/ou organisations et résultent en un processus transactionnel de longue durée [Peltz, 2003].

Définition 2.4. « L'orchestration décrit quant à elle comment les services web peuvent interagir entre eux selon une perspective opérationnelle, avec des structures de contrôle, incluant l'ordre d'exécution des interactions (souvent qualifiée de « logique métier ») » [Pourraz, 2007].

De ces définitions, nous dégageons le rôle principal d'une orchestration, à savoir la description de la logique métier et l'ordre d'exécution des opérations de services.

Proche de la notion de l'orchestration, nous trouvons également la notion de *workflow*. Tandis que l'orchestration vise les services logiciels, le *workflow* peut inclure des tâches humaines en plus des tâches logicielles. Le *Workflow Management Coalition* [WfMC, 2011] est une organisation formée pour définir les standards liés à la gestion des systèmes de *workflow*. Elle présente la définition suivante du *workflow* :

Définition 2.5. Un *workflow* est l'automatisation de tout ou d'une partie d'un processus métier durant lequel des documents, des informations ou des tâches sont transmis d'un participant à un autre pour accomplir une action selon un ensemble de règles de procédure [WfMC, 1999].

Plusieurs langages existent dans la littérature pour décrire les orchestrations de services ou les *workflows*. Ces langages possèdent des structures de contrôle qui ont pour vocation de décrire le déroulement des processus métier. Le projet de recherche *Workflow Pattern Initiative (WPI)* [van der Aalst and ter Hofstede, 1999] a été lancé dans le but de recenser les besoins des langages de description de processus. Le résultat de cette étude est l'identification de vingt structures de contrôle appelées patrons de contrôle de flux de *workflow* (*control-flow patterns*) [van der Aalst et al., 2003]. Dans la suite de ce manuscrit, pour des raisons d'allègement d'écriture, nous utilisons le terme *patrons de workflow* pour désigner les patrons de contrôle de *workflow*. À titre d'exemple, la séquence, le parallélisme, la synchronisation et le choix exclusif sont des patrons de *workflow*.Un deuxième bilan du projet WPI a fait ressortir quarante trois patrons de *workflow* [Russell et al., 2006b]. Ces patrons de *workflow* ont permis d'évaluer le pouvoir d'expression des langages d'orchestration et des langages de description de *workflow* [van der Aalst et al., 2002, van der Aalst and Hofstede, 2002]. Parmi les langages d'orchestration et de description de *workflow*, nous présentons les suivants :

Web Services Business Process Execution Language (WS-BPEL) [TC, 2007] est considéré aujourd'hui comme le langage standard de facto pour la description des orchestrations de services. WS-BPEL est basé sur une grammaire XML, permettant de décrire la logique de contrôle requise pour la coordination entre les participants (services) dans un processus métier. La dernière version de WS-BPEL est WS-BPEL 2.0 et a été standardisée par *Organization for the Advancement of Structured Information Standards (OASIS)* en 2007 [TC, 2007]. Un processus décrit en WS-BPEL comprend deux types d'activités : des activités basiques et des activités structurées. Les activités basiques décrivent les étapes élémentaires du comportement du processus comme l'invocation d'une opération de service (*Invoke*), la réception de message (*Receive*), assigner une variable (*Assign*), etc. Les activités structurées définissent la logique de contrôle du processus. Elles permettent de spécifier la manière dont le processus est exécuté en orchestrant les activités basiques. La séquence, le choix conditionnel (*if*) et le parallélisme (*flow*) sont des exemples

d'activités structurées. Dans une analyse du pouvoir d'expression de WS-BPEL [van der Aalst and Hofstede, 2002], il en ressort que WS-BPEL supporte treize patrons de *workflow* parmi les vingt patrons issus du premier bilan de WPI et dix sept patrons [Russell et al., 2006b] des quarante trois patrons résultant du deuxième bilan.

Process eXtensible Language (PXL) [Verjus et al., 2011] est un langage algébrique basé sur le π-calcul. C'est un langage d'orchestration exécutable qui a été développé essentiellement pour fournir une sémantique opérationnelle formelle (sans ambiguïté). Ceci permet d'avoir une interprétation exacte de l'orchestration par le moteur d'exécution (c'est-à-dire ce qui est décrit correspond exactement à ce qui est exécuté). En effet, WS-BPEL souffre d'un manque de formalisme, d'inconsistance et d'ambiguïté dans sa grammaire [van Breugel and Koshkina, 2006]. Le comité technique de WS-BPEL reconnait le besoin de formalisation pour définir sa sémantique [Committee, 2006]. PXL est le successeur de π-Diapason [Pourraz and Verjus, 2008, Verjus and Pourraz, 2008]. Il introduit de nouveaux mécanismes supportant l'évolution dynamique *non planifiée* des orchestrations de services [Pourraz and Verjus, 2007]. *PXL* est un langage structuré en six couches [Verjus et al., 2011] (voir figure 2.2) :

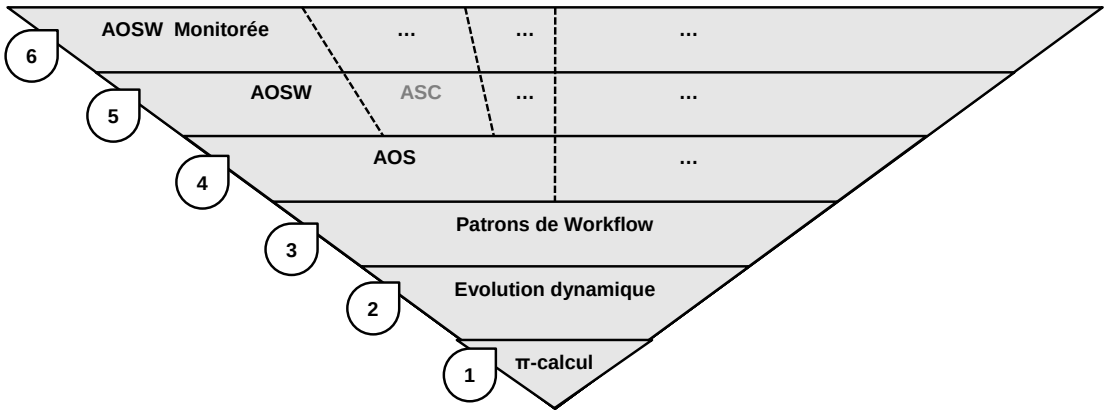


Figure 2.2 – Les différentes couches du langage *PXL* [Verjus et al., 2011]

- la couche π-calcul : c'est la couche noyau qui est exécutable.
- la couche support à l'évolution dynamique : cette couche introduit des mécanismes pour permettre l'évolution dynamique du comportement de l'orchestration à tout moment (y compris au cours de l'exécution).
- la couche patrons de *workflow* : elle décrit quarante deux parmi les quarante trois patrons de *workflow* issus du deuxième bilan de WPI [Russell et al., 2006b]. Cette couche permet de décrire les orchestrations de services plus facilement sans avoir besoin de compétences formelles en π-calcul.
- la couche AOS : elle définit la notion d'invocation d'opérations de service. Cette couche est générique et indépendante de toute implémentation particulière. Elle

permet de décrire d'une manière abstraite les orchestrations de services.

- la couche Architecture Orientée Services Web (AOSW) : cette couche est une déclinaison de la couche AOS pour les services web. Ainsi elle représente une implémentation particulière de l'orchestration de services. D'autres couches (implémentations) au même niveau peuvent être envisagées (voir figure 2.2) comme par exemple l'Architecture Service-Composant (ASC) [Verjus et al., 2011].
- la couche AOSW monitorée : elle définit des mécanismes de supervision des orchestrations des services web décrites en PXL.

Cette architecture en couches permet à l'architecte de modéliser plus facilement les processus avec des construits de plus haut niveau d'abstraction. Chaque couche d'un niveau supérieur est exprimée en termes des couches inférieures. Par conséquent, une orchestration exprimée dans la couche AOSW peut être traduite systématiquement dans la couche π-calcul fournissant ainsi une description formelle du modèle de l'orchestration. La figure 2.3 montre un exemple simplifié d'un processus de production modélisé avec le langage PXL.

```
 1 <process name="production" targetNamespace="http://www.process-box.org/mes">
 2   <sequence>
 3     <invoke service="Gestion_OF" operation="Creer_OF">
 4       <!-- parameters -->
 5     </invoke>
 6     <invoke service="Gestion_OF" operation="Lancer_OF">
 7       <!-- parameters -->
 8     </invoke>
 9     <invoke service="Gestion_OF" operation="Notification_Fin_OF">
10       <!-- parameters -->
11     </invoke>
12       <!-- other invocations -->
13     <compose>
14       <invoke service="Indicateurs_Performance" operation="Calcule_TRS">
15         <!-- parameters -->
16       </invoke>
17       <invoke service="Tracabilite" operation="Exporter_Genealogie_Produit">
18         <!-- parameters -->
19       </invoke>
20       <invoke service="Gestion_Qualite" operation="Qualifier_Produit">
21         <!-- parameters -->
22       </invoke>
23     </compose>
24       <!-- other invocations -->
25   </sequence>
26 </process>
```

Figure 2.3 – Exemple simplifié d'un modèle d'orchestration exprimé en *PXL*

Business Process Modeling Language (BPML) [Assaf Arkin, 2003] est un métalangage pour modéliser les processus métier exécutables [van der Aalst et al., 2002]. C'est un standard proposé par *Business Process Management Initiative* (BPMI) basé sur une syntaxe XML. BPML partage de nombreux concepts communs avec BPEL comme les

activités (basiques et structurées), les variables, les mécanismes de gestion d'exception et de compensation, etc, mais les concepts de BPML comme l'activité *call* (pour instancier un processus et attendre sa terminaison) ou *spawn* (pour instancier un processus sans attendre sa terminaison) n'ont pas d'homologue dans BPEL [Mendling et al., 2003]. Selon l'analyse réalisée dans [van der Aalst et al., 2002], BMPL supporte onze patrons de *workflow* de l'ensemble des vingt patrons présentés dans [van der Aalst et al., 2003].

Les réseaux de Petri (RdP) est un langage graphique de *workflow* basé sur les concepts de places et transitions pour modéliser les processus. Il a été étendu par les couleurs et le temps pour améliorer son expressivité [Jensen, 1995]. Les Réseaux de Petri présentent une sémantique formelle malgré leurs caractères graphique. Ils sont soutenus par une théorie mathématiques permettant l'analyse de leurs propriétés, ils possèdent également un nombre important d'éditeurs graphiques et d'outils de simulation. Les réseaux de Petri présentent quelques limitations dans la modélisation des processus notamment dans les mécanismes d'annulation et de gestion d'instance multiples ainsi que les concepts avancés de synchronisation [van der Aalst and Hofstede, 2002]. Les réseaux de Petri supportent treize des vingt premiers patrons de *workflow* [van der Aalst and Hofstede, 2002].

***Yet Another Workfow Language* (YAWL)** [van der Aalst and ter Hofstede, 2005] est un langage de *workflow* inspiré des réseaux de Petri. Il a été conçu pour couvrir les limitations des réseaux de Petri citées précédemment. YAWL possède une sémantique formelle basée sur les systèmes de transition et offre une représentation graphique de ses concepts. YAWL supporte dix neuf des vingt premiers patrons de *workflow* [van der Aalst and Hofstede, 2002] et vingt neuf patrons des quarante trois patrons de *workflow* issus du deuxième bilan [Russell et al., 2007b]. Les auteurs ont développé une nouvelle version de ce langage *new*YAWL [Russell et al., 2007a] fondée sur les patrons de *workflow*. *new*YAWL est basé sur les réseaux de Petri colorés [Jensen, 1995] permettant de définir avec précision sa sémantique opérationnelle. [Russell et al., 2007b] reportent que *new*YAWL supporte quarante deux sur un total de quarante trois patrons de *workflow*.

Par ailleurs, une caractéristique importante que peu de langages supportent aujourd'hui est *l'adaptation dynamique* des orchestrations de services. On entend par adaptation ou évolution dynamique, la capacité de modifier l'orchestration au cours de son exécution sans avoir besoin de l'arrêter. Nous distinguons l'évolution dynamique planifiée et non planifiée [Cîmpan and Verjus, 2005]. L'évolution dynamique planifiée prévoit d'avance (c'est-à-dire pendant la conception de l'orchestration) les changements à apporter sur l'orchestration. De plus, les instants où une évolution potentielle peut avoir lieu, sont fixés en avance par l'architecte. L'évolution dynamique non planifiée [Pourraz and Verjus, 2007], quant à elle, consiste à modifier le modèle de l'orchestration en cours d'exécution à n'importe quel moment en intégrant n'importe quel scénario d'évolution. Cette caractéristique est native dans le langage PXL [Verjus et al., 2011, Pourraz and Verjus, 2007]. D'autres

travaux [Ben Halima et al., 2008, Qiao et al., 2009] ont proposé des mécanismes d'évolution dynamique (planifiée) pour WS-BPEL moyennant des proxies ou en enrichissant le langage par de nouveaux concepts (voir section 2.4.5). L'évolution dynamique est requise pour remédier à un aléa qui survient au cours de l'exécution quand l'arrêt du processus engendre des pertes/coûts conséquents ; notamment pour les processus critiques, à forte valeur ajoutée et/ou de longues durées. Cependant, pour déclencher l'évolution dynamique d'une orchestration en cours d'exécution, il faut des métriques ou des mécanismes de détection des aléas et des dégradations. La qualité des services et des orchestrations de services représente aujourd'hui un des critères importants pour les clients qu'il faut garantir tout au long de l'exécution de l'orchestration. La dégradation de la qualité des services entrainant potentiellement la dégradation de la qualité de l'orchestration peut donc constituer un paramètre pour déclencher l'évolution dynamique. Dans ce cas, les changement apportés par l'évolution dynamique auront pour but d'améliorer la qualité de l'orchestration.

Dans la section suivante, nous aborderons plus en détails la notion de qualité dans le domaine des AOS.

2.3 Qualité de service

La Qualité de Service (QdS, en anglais *Quality of Service*) est un terme qui est à l'origine apparu dans le domaine des réseaux de télécommunications. À la naissance d'Internet, la fonctionnalité que fournit un réseau de télécommunication (la transmission de données) suivait la politique du « meilleur effort » (*Best Effort–BE*) [Systems, 2010]. Ceci signifie littéralement « Faîtes le du mieux que vous pouvez ». Au fil des années, les technologies et les besoins ont évolué. Un utilisateur exige aujourd'hui une connexion avec un haut niveau de qualité incluant la transmission du contenu multimédia, les services de voix, etc. C'est dans ce contexte que de nombreux travaux se sont intéressés aux mécanismes de gestion de la qualité de service.

Le concept de QdS a été transposé plus tard au domaine des AOS et a fait l'objet de plusieurs travaux de recherche ; notamment dans le domaine de la sélection de services [Taher et al., 2005a, Serhani et al., 2005, Liu et al., 2004] et de la composition de services [Zeng et al., 2003, C. Jaeger, 2007, Rosenberg, 2009]. À ce jour, il n'existe pas de définition standard pour la QdS. Cependant, plusieurs définitions existent dans la littérature [ISO, 1991, Object Management Group, 2008]. Dans le domaine des réseaux de télécommunications, la QdS est définie selon deux points de vues : point de vue utilisateur et point de vue réseau. Du point de vue utilisateur, la QdS est la perception de la qualité qu'il reçoit d'un fournisseur de réseau pour un service ou une application particulière [Park, 2004]. Du point de vue réseau, la QdS est la capacité du réseau à fournir la QdS perçue par l'utilisateur final [Park, 2004]. D'autres définitions existent dans le standard ISO 9004 [ISO, 1991] et dans la spécification *UML Profile for Modelling Quality of Service and Fault Tolerance Characteristic and Mechanisms* [Object Management Group, 2008] respectivement :

Définition 2.6. « L'ensemble des propriétés et des caractéristiques d'un produit ou d'un service qui lui confèrent l'aptitude à satisfaire des besoins explicites ou implicites »

Définition 2.7. « La qualité de service peut être définie comme un ensemble de caractéristiques perceptibles exprimé dans un langage convivial avec des paramètres quantifiables qui peuvent être subjectifs ou objectifs. »

De ces définitions, nous pouvons ressortir deux principaux aspects :

- le premier aspect est que la QdS est représentée par un ensemble de paramètres que nous appellerons *attributs qualité* (voir section 2.3.1),
- le deuxième aspect est le fait qu'on distingue la QdS offerte par le fournisseur de service et la QdS perçue par l'utilisateur final.

2.3.1 Attributs qualité dans les architectures orientées services

Le glossaire de l'ingénierie logicielle du standard IEEE [IEEE, 1990] définit un attribut qualité comme étant :

Définition 2.8. « Une particularité ou une caractéristique qui affecte la qualité d'un objet ».

Dans *UML Profile* [Object Management Group, 2008], le terme caractéristique (*Quality of Service characteristic*) est utilisé pour dénoter un attribut qualité et est défini de la manière suivante :

Définition 2.9. « Les attributs qualité représentent les caractéristiques quantifiables des services. Ils sont spécifiées indépendamment des éléments qu'ils qualifient. L'attribut qualité est le constructeur pour la description des aspects non-fonctionnels tels que la latence, le débit, la disponibilité,... ».

Ceci implique que la qualité des services ou des orchestrations de services est directement liée aux attributs qualité. Nous avons recensé plusieurs attributs qualité [Rosenberg, 2009, Lee et al., 2003, Ran, 2003, Serhani et al., 2005, Taher et al., 2005b, Zeng et al., 2003, Liu et al., 2004, Le Blevec, 2008, Zo et al., 2007, Mani and Nagarajan, 2002] divisés principalement en trois catégories [Ran, 2003] :

Attributs qualité liés à l'exécution

Temps d'exécution (q_{ex}) **:** [Zeng et al., 2003, Liu et al., 2004, Rosenberg, 2009] étant donné une opération o d'un service s, le temps d'exécution est le temps mis par le serveur pour traiter la requête (voir Fig. 2.4) :

$$q_{ex}(s, o) = t_2 - t_1$$

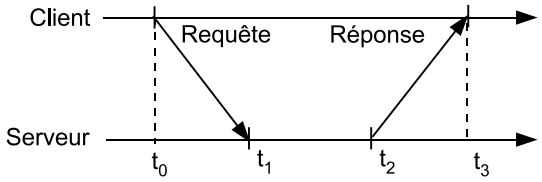


Figure 2.4 – Chronologie d'invocation de service

où t_1 est l'instant à partir duquel le serveur reçoit la requête du client et t_2 l'instant à partir duquel il termine le traitement de la requête et envoie la réponse au client.

Latence (q_{lat}) : [Zeng et al., 2003, Liu et al., 2004, Rosenberg, 2009] la latence d'une opération o d'un service s est le temps consommé par la requête dans le réseau entre le client et le serveur :

$$q_{lat}(s,o) = (t_1 - t_0) + (t_3 - t_2)$$

où t_0 représente l'instant d'envoi de la requête par le client et t_3 l'instant de réception de la requête.

Temps de réponse (q_{rt}) : [Serhani et al., 2005, Taher et al., 2005b, Rosenberg, 2009] le temps de réponse d'une opération o d'un service s est le temps requis pour l'envoi de la requête jusqu'à ce qu'elle soit reçue par le client :

$$q_{rt}(s,o) = t_3 - t_0$$

Il peut être calculé aussi comme étant la somme du temps d'exécution et de la latence :

$$q_{rt}(s,o) = q_{ex}(s,o) + q_{lat}(s,o) \ .$$

Disponibilité (q_{av}) : [Serhani et al., 2005, Taher et al., 2005b, Zeng et al., 2003, Lee et al., 2003] la disponibilité d'une opération o d'un service s est la probabilité que cette opération de service soit opérationnelle pendant une période de temps donnée. Elle peut être mesurée de la manière suivante :

$$q_{av}(s,o) = \frac{up\ time}{up\ time + down\, time}$$

où *up time* est le temps pendant lequel l'opération de service est opérationnel et *down time* le temps pendant lequel l'opération ne répond pas.

Fiabilité (q_{rel}) : [Le Blevec, 2008, Zeng et al., 2003, Ran, 2003, Lee et al., 2003] la fiabilité d'une opération o d'un service s est la probabilité que l'opération fonctionne avec succès pendant une période donnée dans les conditions spécifiées. Une opération de service répond

avec succès s'il n'y a pas eu de défaillances pendant son exécution. [Cortellessa and Grassi, 2007] distingue trois types de défaillances :

- défaillance régulière : c'est une défaillance qui se manifeste comme une valeur inattendue pour toute réponse de l'opération de service ;
- défaillance entraînant une panne (*crash failure*) : c'est une défaillance qui engendre l'arrêt immédiat de l'exécution de l'opération de service. Les systèmes qui subissent ce type de défaillance sont connus comme des systèmes de « défaillance-et-arrêt » (*fail-and-stop systems*) ;
- défaillance répétitive : c'est une défaillance qui empêche l'opération de service de produire toute réponse (correcte ou incorrecte).

Ces trois types de défaillances peuvent être partagés en défaillances réparables ou non réparables [Cortellessa and Grassi, 2007]. Les défaillances réparables ne nécessitent pas le redémarrage du système, contrairement aux défaillances non réparables. Dans la suite, nous adoptons le type de défaillance entraînant une panne et non réparable. Ainsi, la fiabilité peut être mesurée comme suit :

$$q_{rel}(s, o) = \frac{n_s}{n_t}$$

où n_s représente le nombre de requêtes effectuées avec succès et n_t est le nombre total de requêtes.

Accessibilité (q_{acs}) : [Le Blevec, 2008, Taher et al., 2005b, Mani and Nagarajan, 2002, Lee et al., 2003] c'est la capacité du service de satisfaire les requêtes des clients. L'accessibilité peut être exprimée comme une mesure de probabilité indiquant le taux de succès d'une instanciation réussie du service à un moment donné. Il se peut qu'une opération de service soit disponible mais non accessible du fait du nombre de requêtes importants.

Coût (q_c) : [Serhani et al., 2005, Taher et al., 2005b, Zeng et al., 2003, Liu et al., 2004, Rosenberg, 2009] c'est la valeur monétaire associée à l'invocation d'une opération o d'un service s. Généralement, le coût est spécifié par invocation d'une opération de service. Toutefois, des stratégies de facturation avancées peuvent être définies dans le contrat comme par exemple la facturation par palier (80 euros pour les 1000 premières requêtes et 50 euros pour les 500 requêtes suivantes).

Débit (q_{th}) : [Taher et al., 2005b, Ran, 2003, Rosenberg, 2009] c'est le nombre de requêtes accomplies pendant une période donnée. On trouve aussi le débit maximum (ou capacité [Ran, 2003]) qui est le nombre maximum de requêtes simultanées servies pendant une période de temps donnée avec les performances garanties. Le débit peut être calculé de la manière suivante [Rosenberg, 2009] :

$$q_{th}(s, o) = \frac{r}{t_d - t_f}$$

où r est le nombre de requêtes accomplies pendant l'intervalle de temps $[t_d - t_f]$

Performance (q_{per}): [Taher et al., 2005b,Ran, 2003,Le Blevec, 2008,Lee et al., 2003] c'est la mesure de la rapidité pour qu'une invocation d'une opération de service soit complète. Elle est fonction du temps de réponse, de la latence et du débit. Dans [Taher et al., 2005b], la performance est calculée en divisant la somme de n temps de réponse (reportés par n consommateurs du service) par le débit $q_{th}(s, o)$ du service à un instant donné :

$$q_{per}(s, o) = \frac{\sum_{i=1}^{n} q_{rt_i}(s, o)}{q_{th}(s, o)}$$

où $q_{rt_i}(s, o)$ est le temps de réponse reporté par le client i.

Réputation (q_{rep}) : [Serhani et al., 2005, Zeng et al., 2003, Liu et al., 2004, Rosenberg, 2009] est une mesure de la fidélité (en anglais *trustworthiness*) du service. Elle dépend de l'expérience des clients utilisant le service. Différents clients peuvent avoir différents avis sur le même service. Les avis (retours) des clients peuvent être représentés par des notes dans un intervalle par exemple [1,5] (1 est la plus mauvaise note et 5 est la meilleure). Dans ce cas la réputation peut être calculé comme le classement moyen donné au service par les clients :

$$q_{rep}(s) = \frac{\sum_{i=1}^{n} R_i}{n}$$

où R_i est le classement donné par un client final du service s et n le nombre de fois que le service a été classifié.

Adaptabilité (q_{sc}) : [Rosenberg, 2009,Le Blevec, 2008,Taher et al., 2005b,Ran, 2003,Lee et al., 2003] c'est la capacité à augmenter la capacité de traitement des systèmes informatiques par les fournisseurs de services et l'aptitude du système à traiter davantage d'opérations ou de transactions dans une période donnée [Le Blevec, 2008, Taher et al., 2005b, Ran, 2003]. L'adaptabilité est étroitement liée à la performance et au débit. [Rosenberg, 2009] définit l'adaptabilité comme étant la capacité d'un service de préserver les exigences en termes du temps de réponse et de débit quand les requêtes des clients augmentent. Dans ce contexte, l'adaptabilité d'une opération o d'un service s peut être calculée par la formule suivante [Rosenberg, 2009] :

$$q_{sc}(s, o) = \frac{\frac{1}{n}\sum_{i=1}^{n} q_{rt_i}(s, o)}{\frac{1}{m}\sum_{i=1}^{m} q_{rt(Throughput)_i}(s, o)} \quad .$$

Le principe de cette formule est de calculer le quotient de la moyenne du temps de réponse $q_{rt}(s, o)$ de n requêtes (réalisées dans les conditions normales) par la moyenne du temps de réponse $q_{rt(Throughput)}(s, o)$ durant un test de débit de m requêtes effectuées en parallèles. Dans le cas où le service n'est pas adaptable, le temps de réponse moyen durant le test de débit de m requêtes simultanées (dénominateur) va augmenter ce qui conduit à

diminuer l'adaptabilité. Cette dernière est définie sur l'intervalle [0,1]. On dit qu'un service est adaptable si $q_{sc}(s) > 0.5$ et non adaptable dans le cas contraire [Taher et al., 2005b].

Robustesse (q_{rob}) : [Ran, 2003, Le Blevec, 2008, Rosenberg, 2009, Lee et al., 2003] c'est le degré pour lequel le service peut fonctionner correctement en présence de paramètres d'entrée incomplets, invalides ou en présence de conflit dans ces derniers.

Réglementation (q_{reg}) : [Ran, 2003, Le Blevec, 2008] c'est le degré de conformité du service avec le contrat établi entre le client et le fournisseur, les règles, les lois, le respect des standards (par exemple SOAP, UDDI, WSDL pour les services web), etc.

Précision (q_{acr}) : il y a un désaccord de la communauté concernant la définition de cet attribut qualité : certains [Rosenberg, 2009, Ben Halima et al., 2008] le définissent comme étant le taux de succès produit par le service, d'autres [Ran, 2003, Le Blevec, 2008, Lee et al., 2003] le définissent comme étant le taux d'erreurs produites par le service. La première définition revient à celle de l'attribut qualité de fiabilité. Par conséquent, nous considérons que la deuxième définition est plus adéquate (notons que [Rosenberg, 2009, Ben Halima et al., 2008] n'ont pas considérer l'attribut qualité de fiabilité). Dans ce cas, la précision d'une opération o d'un service s peut être obtenue en relevant le nombre d'erreurs produites n_f du nombre total d'invocations n_t réalisées sur un intervalle de temps donnée :

$$q_{acr}(s, o) = \frac{n_f}{n_t} \ .$$

Tableau 2.1 – Récapitulatif des attributs qualité liés à l'exécution

Attribut qualité	**Symbole**	**Dimension**	**Unité**	**Déterministe**
Temps d'exécution	q_{ex}	desc	temps	non
Latence	q_{lat}	desc	temps	non
Temps de réponse	q_{rt}	desc	temps	non
Disponibilité	q_{av}	asc	pourcentage	non
Fiabilité	q_{rel}	asc	pourcentage	non
Accessibilité	q_{acs}	asc	pourcentage	non
Coût	q_c	desc	valeur monétaire	oui
Débit	q_{th}	asc	pourcentage	non
Performance	q_{per}	asc	pourcentage	non
Réputation	q_{rep}	définie par le client	définie par le client	non
Adaptabilité	q_{sc}	asc	pourcentage	non
Robustesse	q_{rob}	asc	pourcentage	non
Réglementation	q_{reg}	asc	pourcentage	non
Précision	q_{acr}	desc	pourcentage	non

asc : ascendant ; desc : descendant

Attributs qualité liés à la sécurité

Cette catégorie d'attributs qualité concerne les mécanismes de sécurité et de confidentialité implémentés. Dans cette catégorie, nous pouvons recenser les attributs qualité suivants :

Authentification (q_{auth}) : [Haas and Brown, 2004, Booth et al., 2004, Ran, 2003] elle permet de vérifier l'identité de l'utilisateur et d'assurer que seuls les utilisateurs autorisés ont accès aux ressources.

Autorisation (q_{auto}): [Haas and Brown, 2004, Booth et al., 2004, Ran, 2003] elle permet de contrôler le droit d'accès aux ressources appropriées. Souvent l'autorisation est liée à l'authentification. Une fois l'utilisateur authentifié, il peut être autorisé à effectuer différents types d'accès.

Confidentialité (q_{conf}) : [Ran, 2003, Booth et al., 2004] c'est la protection contre l'interception des informations sensibles et des transactions. La méthode de protection contre ce type de menace est le cryptage de données.

Attribut qualité liés aux transactions

Cette catégorie d'attribut qualité est issue du domaine des bases de données. Les quatre principaux attributs d'une transaction de données [Borderie, 2006] sont l'atomicité, la cohérence, l'isolation et la durabilité souvent connues sous l'acronyme des propriétés ACID :

Atomicité (q_{ato}) : [Durate-Amaya, 2007] elle est liée aux mécanismes de compensation et d'annulation. Toutes les opérations d'une transaction sont exécutées ou aucune. Elle est assurée par un mécanisme qui compense les effets de la transaction lorsque celle-ci doit être annulée.

Cohérence (q_{coh}) : [Durate-Amaya, 2007,Borderie, 2006] elle signifie que les modifications apportées à la base de données sont valides (en accord avec les contraintes d'intégrité). Si à l'issue d'une transaction, une contrainte d'intégrité est violée, la transaction est annulée.

Isolation (q_{iso}) : [Durate-Amaya, 2007, Borderie, 2006] elle signifie que les transactions exécutées en concurrences ne doivent pas interférer entre elles. Les effets non validés d'une transaction ne sont pas visibles pour les autres.

Durabilité (q_{dur}) : [Durate-Amaya, 2007, Borderie, 2006] elle est liée à la persistance des données. Si une transaction est validée, les changements qu'elle a provoqués doivent survivre aux défaillances.

Synthèse : nous avons présenté dans cette section plusieurs attributs qualité dans le domaine des AOS divisés en trois catégories à savoir les attributs qualité liés *(i)* à l'exécution, *(ii)* à la sécurité et *(iii)* aux transactions. Les travaux de cette thèse adressent particulièrement les attributs qualité liés à l'exécution. En effet, rappelons que notre objectif est de préserver la qualité des orchestrations de services au cours de leurs exécutions et d'effectuer les actions nécessaires quand une dégradation de la qualité survient. Les attributs qualité liés à la sécurité et aux transactions sont statiques. Par conséquent, nous considérons qu'ils ne sont pas pertinents pour l'illustration de notre approche. Le tableau 2.1 récapitule les attributs qualité liés à l'exécution. La colonne « dimension » spécifie si l'attribut qualité a une dimension ascendante (asc) ou descendante (desc). Une dimension ascendante (respectivement descendante) signifie que plus la valeur de l'attribut qualité est élevée, meilleur (respectivement médiocre) il est. La colonne « déterministe » indique si l'attribut qualité est déterministe ou non. Un attribut qualité est non déterministe s'il change de valeur d'une instance de service à une autre, comme par exemple le temps de réponse. Nous pouvons observer d'après ce tableau que la majorité des attributs qualité liées à l'exécution sont non déterministes. Ce caractère non déterministe (dynamique) de certains attributs qualité engendre des conséquences sur les relations clients-fournisseurs de services. En effet, les clients ont besoin d'une garantie de la qualité perçue lors de l'utilisation des services. Ceci est concrétisé à travers les contrats de niveau de service où le client et le fournisseur de service, ensemble, signent un contrat qui spécifie entre autres les conditions d'utilisation et le niveau de qualité garanti.

Dans la section suivante, nous allons présenter les travaux réalisés sur les contrats de niveau de service en se focalisant sur l'aspect qualité.

2.3.2 Contrats de niveau de service

Le contrat de niveau de service, en anglais *Service Level Agreement (SLA)*, est un contrat établi entre l'utilisateur et le fournisseur de service dans lequel on définit la qualité de service requise [Castellanos et al., 2003, Szydlo and Zielinski, 2008]. Il formalise, avec l'accord des deux parties signataires, les responsabilités, les priorités, les garanties : ce que l'on pourrait définir comme « niveau de service » [Keller and Ludwig, 2003]. À titre d'exemple, il peut permettre de spécifier la disponibilité du service, le taux de fiabilité, la performance ou encore le coût d'utilisation et les pénalités en cas de violation du contrat.

Plusieurs travaux dans la littérature ont porté sur la formalisation des contrats de niveau de service (SLA) [Keller and Ludwig, 2003, Sahai et al., 2002, Tosic et al., 2002b]. *Web Service Level Agreement* (WSLA) [Ludwig et al., 2003, Keller and Ludwig, 2003] proposé par *International Business Machines* (IBM) est un langage XML de spécification, de monitoring et de gestion des contrats de niveau de service. Un contrat de niveau de service exprimé en WSLA contient trois sections. La première section définit les parties signataires du contrat à savoir le client, le fournisseur et éventuellement une partie tierce chargée de surveiller le contrat. La deuxième section du contrat est consacrée à la descrip-

```
1   <wsla:Parties>
2     <wsla:ServiceProvider name="EnterpriseA" />
3     <wsla:ServiceConsumer name="EnterpriseB" />
4   </wsla:Parties>
5   <wsla:ServiceDefinition name="ProductionRequestsManagement">
6         <wsla:Operation name="CreateProductionRequest">
7               <wsla:SLAParameter name="ResponseTime" type="float" unit="sec">
8                     <wsla:Metric>ResponseTimeMetric</wsla:Metric>
9               </wsla:SLAParameter>
10        </wsla:Operation>
11  </wsla:ServiceDefinition>
12  <wsla:Obligations>
13        <wsla:ServiceLevelObjective>
14              <wsla:Obliged>EnterpriseA</wsla:Obliged>
15              <wsla:Validity>
16                    <wsla:Start>2011-01-01T12:00:00</wsla:Start>
17                    <wsla:End>2011-12-31T12:00:00</wsla:End>
18              </wsla:Validity>
19              <wsla:Expression>
20                    <wsla:Predicate xsi:type="wsla:LessEqual">
21
22        <wsla:SLAParameter>ResponseTime</wsla:SLAParameter>
23                          <wsla:Value>30</wsla:Value></wsla:Predicate>
24              </wsla:Expression>
25              <wsla:EvaluationEvent>NewValue</wsla:EvaluationEvent>
26        </wsla:ServiceLevelObjective>
27    </wsla:Obligations>
28  </wsla:SLA>
```

Figure 2.5 – Extrait d'un contrat de niveau de service exprimé avec le langage WSLA

tion du service en question. Les paramètres sur lesquels portent le contrat sont spécifiés dans cette section avec leurs métriques, les méthodes pour les mesurer et/ou les calculer. À un paramètre du contrat, peut être associé un déclencheur ou un calendrier permettant de définir une ou plusieurs dates auxquelles une vérification des conditions, un calcul ou une mesure des métriques ont lieu. La troisième section du contrat est dédiée à la définition des obligations (*Service Level Objectives–SLOs*) et des actions de garantie. Une obligation décrit le niveau de service garanti des paramètres définis dans le contrat pour une période particulière. Quant aux actions de garantie, elles spécifient ce qui peut être fait en cas de non respect du contrat comme par exemple le paiement de pénalités ou la notification du fournisseur de service de la violation qui a eu lieu afin qu'il puisse prendre les actions nécessaires.

La figure 2.5 montre un exemple simplifié d'un contrat de niveau de service entre une entreprise A et un entreprise B représentant respectivement le fournisseur et le consommateur de service (lignes 1-4). Le contrat porte sur le temps de réponse de l'opération *CreateProductionRequest* appartenant au service *ProductionRequestsManagement* (lignes 5-11). Une obligation (SLO) est définie pour exprimer que le fournisseur de service (entreprise A) s'engage, pour une période d'un an (lignes 12-18), à ce que le temps de réponse de l'opération de service n'excède pas trente secondes (lignes 19-23). L'élément *Evaluation-*

Event indique que l'obligation doit être vérifiée chaque fois qu'il y a une nouvelle valeur du temps de réponse (ligne 25).

Web Service Management Language (WSML) [Sahai et al., 2002] est un autre langage XML, développé par *Hawlett-Packard (HP)* pour formaliser les contrats de niveau de service pour les services web. WSML est compatible avec WSDL dans le sens où les paramètres, sur lesquels porte le contrat, peuvent être directement associés aux opérations de service déjà définies dans l'interface WSDL. On retrouve les concepts du WSLA dans WSLM à savoir la validité du contrat, les parties signataires et les obligations. Cependant, WSLM ne supporte pas la notion de partie tierce (chargée de surveiller le contrat) comme c'est le cas avec WSLA.

Web Service Offering Language (WSOL) [Tosic et al., 2002b, Tosic et al., 2002a] est un langage XML pour la spécification formelle de multiples classes d'un service. Une classe d'un service est déterminée par des contraintes sur les attributs qualité, les modalités de paiement, le coût, etc. Ceci permet d'offrir le même service avec différentes classes selon le besoin d'utilisation. La principale idée du langage est de décrire plusieurs offres du même service. Il permet de spécifier des contraintes fonctionnelles, des contraintes sur des attributs qualité (contraintes non-fonctionnelles), les droits d'accès, le coût et les pénalités, et les relations avec les autres offres du même service. Les contraintes fonctionnelles permettent de définir les conditions qui doivent être satisfaites pour que l'opération de service fonctionne correctement. WSOL permet de spécifier les pré-conditions, les post-conditions et les conditions futures. Les conditions futures sont un nouveau concept qui consiste à évaluer la condition à une date spécifique ou après une certaine période de la fin de l'exécution de l'opération. Ceci est différent des post-conditions qui sont évaluées à la fin de l'exécution de l'opération. Un exemple de condition future est la confirmation de livraison d'une marchandise achetée à l'aide d'un service *web*. Les contraintes non-fonctionnelles décrivent les attributs qualité comme le temps de réponse, la fiabilité, la disponibilité, etc. Elles permettent de vérifier si les valeurs des attributs qualité surveillés respectent les limites spécifiées. Toute contrainte dans WSOL est une expression booléenne qui spécifie une condition qui doit être vérifiée avant et/ou après l'exécution de l'opération de service. Par exemple, le temps de réponse maximum autorisé est 50ms. Les contraintes sur les droits d'accès spécifient les conditions pour qu'un consommateur ait le droit d'invoquer une opération de service particulière. À titre d'exemple, supposons qu'un service donné possède plusieurs opérations. Une offre de service peut autoriser l'invocation d'une seule opération de service particulière qui remplit le besoin du consommateur. Enfin, la spécification des relations entre plusieurs offres du même service permet, dans le cas où l'offre du service choisi par le consommateur ne peut pas satisfaire une contrainte, de lui proposer une autre offre de service.

Synthèse : nous avons présenté dans cette section les travaux relatifs à la spécification des contrats de niveaux de service. Ces travaux permettent de spécifier les exigences des consommateurs de services en termes de contraintes sur les attributs qualité. Il convient par la suite de vérifier le respect de ces contraintes et d'assurer leur satisfaction lors de l'utilisation des services, ceci est le sujet des travaux que nous allons présenter dans la suite de ce chapitre.

2.4 Qualité des orchestrations de services

Dans un contexte de sélection de services, les clients choisissent les services répondant à leurs besoins fonctionnels et non fonctionnels. Avec la prolifération des services équivalents (qui offrent les mêmes fonctionnalités), les clients ont tendance à choisir les services satisfaisant au mieux leurs exigences en termes d'attributs qualité. Au delà de la qualité des services pris indépendamment au moment de la sélection, les clients peuvent avoir des contraintes globales sur les attributs qualité portant sur l'orchestration [Canfora et al., 2005a, Menascé et al., 2010]. Par exemple, le temps de réponse global de l'orchestration ne doit pas dépasser une certaine limite. Ce genre de contrainte peut être formalisé à travers les contrats de niveau de service (SLA) portant sur l'orchestration globale. Dans ce contexte, nous présentons dans la section 2.4.1 les travaux liés à la composition des contrats.

D'autre part, dans un contexte de supervision des orchestrations, une déviation de la qualité d'un service peut être compensée par une meilleure qualité d'autres services ou par la structure logique de l'orchestration. Par exemple, considérons deux services A et B dont les de temps de réponse sont respectivement $10s$ et $15s$. Supposons que ces services sont exécutés en parallèle suivis d'un service C qui ne sera exécuté qu'après la fin d'exécution des services A et B. Dans ce cas, une dégradation du temps de réponse du service A, ne dépassant pas les $5s$, n'affectera pas le temps de réponse globale de l'orchestration. Il est donc plus judicieux de raisonner non seulement sur la qualité des services mais aussi sur la qualité globale de l'orchestration. Afin de superviser la qualité globale des orchestrations, il est nécessaire d'évaluer les attributs qualité des orchestrations en partant des valeurs des attributs qualité des services qui leur composent. Dans ce sens, nous pouvons diviser les approches permettant de répondre à cet objectif en deux catégories : approches basées sur des modèles probabilistes (section 2.4.2) et approches basées sur des règles d'agrégation de patrons de *workflow* (section 2.4.3).

2.4.1 Composition des contrats de niveau de service

Nous avons vu dans la section 2.3.2 que WSLA, WSML et WSOL sont des langages de formalisation et de gestion des contrats de niveau de service entre un utilisateur et un fournisseur de service. Ces travaux se focalisent uniquement sur la relation bi-latérale entre le client et le fournisseur et ne gèrent pas les contrats (SLA) entre un utilisateur d'un service composite (orchestration de services, elle même exposée sous la forme d'un service) et les

fournisseurs des services (atomiques) impliqués dans ce service composite. [Ludwig and Franczyk, 2008, Ludwig et al., 2009] proposent *COmposite Sla MAnagement (COSMA)*, une approche de gestion des contrats composites. L'approche consiste à intégrer tous les contrats des services, que le fournisseur de l'orchestration de services a établi avec les fournisseurs des services composants, en un seul contrat composite. Ceci permet à l'utilisateur de l'orchestration de ne faire face qu'à un seul contrat (composite) au lieu de plusieurs (un contrat par service impliqué dans l'orchestration). Afin de déterminer les obligations (SLO) portant sur les paramètres du SLA composite, le modèle de l'orchestration est décomposé en patrons de *workflow*. Ensuite, des règles d'agrégation [Jaeger et al., 2004] (voir section 2.4.3) sont utilisées pour calculer le paramètre sur lequel porte le contrat composite à partir de ceux des contrats atomiques.

[Wetzstein et al., 2008] proposent une approche pour gérer la composition des contrats de niveau de service basée sur les indicateurs de performance métier (*Key Performance Indicator*–KPI). L'approche consiste en plusieurs étapes. Dans la première étape, les indicateurs de performance métiers et les besoins des clients de l'orchestration (traduit par des obligations) sont identifiés. Ensuite, pendant la phase de configuration, les relations entre les indicateurs de qualité et les obligations des fournisseurs de services (candidats pour l'orchestration) sont cartographiées. Le but est d'identifier quel paramètre, sur lequel porte le contrat, influence les indicateurs de performance métiers. La figure 2.6 montre un exemple de relations entre un indicateur de performance métier et les paramètres sur lesquels porte le contrat d'un processus de traitement des ordres d'achat. Considérons par exemple l'indicateur de performance de ce processus métier : le pourcentage des ordres d'achat « complets » (c'est-à-dire tous les articles commandés sont délivrés) et « à temps » est supérieur à 90%. La condition « complet » peut être satisfaite si le fournisseur possède tous les articles requis ou s'ils sont disponibles en stock. La condition « à temps » est traduite par la durée du processus jusqu'à la réception des articles par le client. Cette durée est associée au temps de livraison et au délai d'expédition qui sont garantis par des contrats établis avec les services de livraison et d'expédition. Le temps d'exécution des invocations des opérations de service doit être également pris en compte. En outre, la disponibilité de l'infrastructure peut impacter cette durée. L'étape suivante consiste à s'assurer que la valeur exigée de l'indicateur de performance (portant sur le processus métier) peut être atteinte avec les obligations des services partenaires et les performances de l'infrastructure (moteur d'exécution des orchestrations, serveurs, bases de données, etc). Pour ce faire, les obligations des services partenaires sont agrégées [Jaeger et al., 2005] et comparées aux valeurs des indicateurs de performance. Dans la dernière étape, les auteurs proposent d'utiliser un système de supervision (du côté client) pour surveiller les contrats au cours de l'exécution. En outre, ils proposent d'utiliser un outil de supervision des processus métiers (*Business Activity Monitoring*) permettant au fournisseur de service de surveiller les indicateurs de performance métiers. Ceci permet d'identifier les sources de dégradation des indicateurs de performance en se basant sur la cartographie (des relations

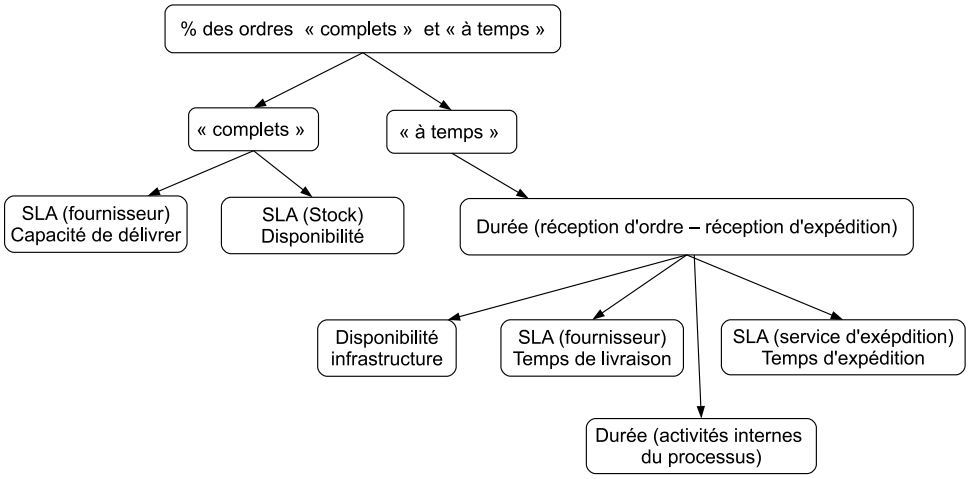


Figure 2.6 – Exemple de relations entre un indicateur de performance et les paramètres du SLA

entre indicateurs de performance et obligations) réalisée précédemment. Pour l'exemple d'indicateur de performance cité précédemment, il est possible d'analyser si la cause de la dégradation est due au service de livraison ou au service d'expédition, etc.

Les travaux présentés précédemment sur la composition des contrats de niveau de service considèrent des contraintes déterministes sur les attributs qualité. À titre d'exemple, le temps de réponse d'une opération de service ne doit pas excéder une certaine valeur fixée pendant une période donnée. Ceci n'est pas cohérent avec la nature non déterministe des attributs qualité liés à l'exécution (voir section 2.3.1). [Rosario et al., 2008, Rosario et al., 2009a, Rosario et al., 2010] considèrent que les contraintes déterministes ne sont pas réalistes dans la pratique. Au lieu de considérer des bornes strictes sur les attributs qualité, les auteurs proposent des contrats statistiques appelés aussi des contrats probabilistes. Dans de tels contrats, les attributs qualité sont modélisés avec des distributions de probabilités et les bornes strictes sont remplacées par des obligations probabilistes. Par exemple, la probabilité que le temps de réponse soit inférieur à un seuil x est $y\%$. Ainsi, établir un contrat statistique sur l'orchestration consiste donc à agréer les distributions de probabilités des attributs qualité en question [Rosario et al., 2009b] (voir section 2.4.4). De cette manière, la surveillance du contrat devient plus flexible : par exemple, le fait de franchir une seule fois le seuil limite ne devrait pas être considéré comme une violation.

2.4.2 Approches basées sur les modèles probabilistes

Dans [Zhong and Qi, 2006], les auteurs proposent une approche basée sur les réseaux de Petri pour prédire la fiabilité des orchestrations de services avant l'exécution. Le principe de l'approche est résumé dans la figure 2.7a et comprend quatre étapes. La première étape consiste à transformer le modèle de l'orchestration (décrit en WS-BPEL) en réseaux de Petri. Il existe plusieurs types de réseaux de Petri. Le type choisi dans cette approche

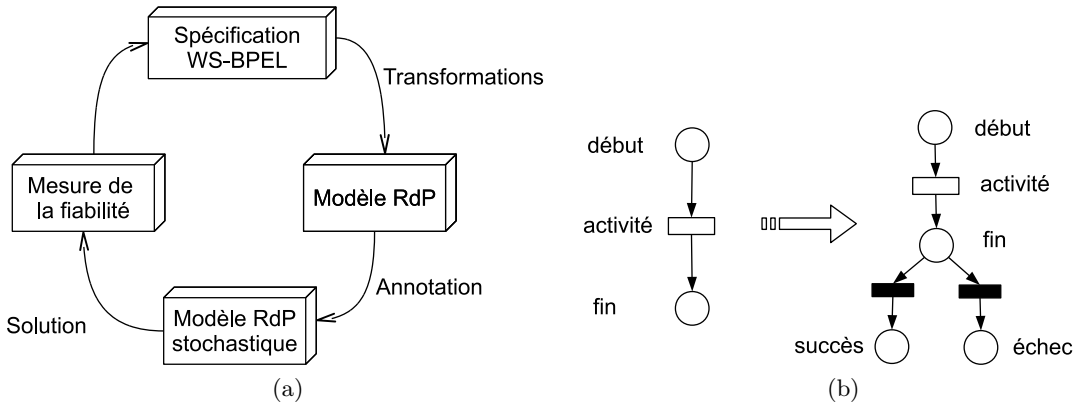


Figure 2.7 – (a) Approche de prédiction de la fiabilité ; (b) Modélisation de la fiabilité

est le *Stochastic Reward Nets (SRN)* [Ciardo et al., 1993]. Les activités basiques de WS-BPEL (*Invoque, Receive,...*) sont transformées en transitions temporisées et les activités structurées (choix, parallélisme,...) sont traduites par les transitions immédiates ainsi que les règles de tirage des jetons. Ensuite, le modèle obtenu en réseaux de Petri est annoté : chaque transition temporisée est annotée avec le temps de réponse de l'activité. Les transitions immédiates sont annotées avec les probabilités des branches conditionnelles tandis que la fiabilité est prise en compte différemment. Pour représenter la fiabilité, le modèle du réseau de Petri est étendu comme le montre la figure 2.7b ; chaque transition immédiate du modèle initial (issu de la transformation du modèle en WS-BPEL) est remplacée par deux transitions immédiates suivies de deux places qui représentent le succès et l'échec de l'activité. Ces deux transitions portent des poids qui sont respectivement la fiabilité et le taux d'échec de l'activité. La dernière étape consiste à résoudre le modèle du réseau de Petri stochastique et de déterminer les valeurs des attributs qualité de l'orchestration. Pour ce faire, les auteurs ont eu recours au Stochastic Petri Net Package (SPNP) [Hirel et al., 2000]. L'inconvénient de cette approche est que le nombre d'attributs qualité supportés est limité. D'autre part, les auteurs supposent que les distributions de probabilités du temps de réponse et de la fiabilité sont des distributions exponentielles négatives ce qui n'est pas le cas en pratique, notamment pour le temps de réponse [Rosario et al., 2008, Kattepur et al., 2010]. En effet, [Rosario et al., 2008] ont mesuré le temps de réponse de vingt milles invocations pour six services web différents. Les résultats ont montré que la distribution de probabilités la plus convenable correspond à la distribution t de *Student*. Les auteurs de cette étude mentionnent aussi que les distributions *Gamma* et *Log-Logistic* correspondent raisonnablement aux données mesurées.

[Gallotti et al., 2008] présentent un outil appelé *Activity Diagram TO Prism models (ATOP)*. Il illustre une méthode de vérification énumérative probabiliste *(probabilistic model-checking)* pour estimer les valeurs des attributs qualité d'une orchestration de services. L'objectif est de vérifier la concordance de la qualité des services avec les besoins spécifiés par l'architecte sur l'orchestration globale. L'approche présentée traite la phase

de conception de l'orchestration et se base sur le diagramme d'activité d'UML pour représenter le modèle de l'orchestration. En particulier, MARTE [Object Management Group, 2007] – un profil UML conçu pour la spécification des propriétés non-fonctionnelles des systèmes logiciels – est utilisé pour annoter les activités et les branches sortantes des noeuds décisionnels. Le principe d'annotation est similaire à l'approche présentée précédemment. Les attributs qualité pris en compte sont restreints aux temps de réponse et à la fiabilité. Le modèle annoté est ensuite transformé en un type de modèle de Markov selon sa nature et les attributs qualité à analyser. Le modèle des chaînes de Markov discrètes *(Discrete Time Markov Chains – DTMC)* est utilisé lorsque seul l'attribut qualité de fiabilité est considéré ; le temps de réponse n'est pas pris en compte par le modèle DTMC. En outre, le modèle DTMC ne permet pas de modéliser le parallélisme. Pour supporter les modèles d'orchestrations contenant des structures de parallélisme, il est nécessaire d'utiliser un modèle de processus de décision de Markov (*Markov Decision Process– MDP)* qui permet de modéliser toutes les invocations entrelacées possibles. Enfin, le modèle des chaînes de Markov continues *(Continuous Time Markov Chains–CTMC)* permet d'analyser le temps de réponse global de l'orchestration. Là encore, le temps de réponse est supposé suivre une distribution exponentielle négative.
Les auteurs exploitent les logiques temporelles probabilistes *(Probabilistic Computation Tree Logic – PCTP)* pour spécifier les exigences de l'orchestration en termes de QdS. Le modèle de vérification énumérative prend en entrée le modèle de Markov et les propriétés spécifiées, et produit les résultats d'analyse correspondants. L'architecte pourra ensuite vérifier si le modèle d'orchestration satisfait les besoins en QdS.

Dans le même contexte, [Sato and Trivedi, 2007] présentent une approche analytique basée sur un modèle de Markov CMTC. Comme nous l'avons souligné précédemment, ce modèle CMTC permet de décrire les attributs qualité de fiabilité et du temps de réponse. L'approche proposée considère les tentatives de ré-invocation comme stratégie de tolérance aux fautes. La figure 2.8 décrit la modélisation de l'échec d'une invocation de service avec tentative de ré-invocation. Supposons qu'un service possède un temps de réponse moyen λ^{-1} et une probabilité de succès R (c'est-à-dire de fiabilité). La transition vers l'état « succès » est annotée avec une vitesse de transition $\lambda.R$ tandis que l'autre transition a une vitesse de transition $\lambda.(1-R)$. En cas d'échec de la première invocation du service, une nouvelle tentative d'invocation peut réussir avec une probabilité C et échouer avec une probabilité $1-C$. Les probabilités de transitions correspondantes sont respectivement $\lambda_{inv}.C$ et $\lambda_{inv}.(1-C)$; avec λ_{inv}^{-1} le temps moyen dû à la nouvelle tentative d'invocation. Les auteurs présentent aussi une analyse de la sensibilité des services impliqués dans l'orchestration. L'analyse permet de détecter le service qui impacte négativement les valeurs des attributs qualité de l'orchestration globale. Par exemple, la sensibilité d'un service s par rapport à la fiabilité est obtenue en calculant la dérivée partielle de la fiabilité de l'orchestration globale par rapport à la fiabilité du service s. En calculant la sensibilité de tous les services impliqués dans l'orchestration, le service ayant la sensibilité la plus

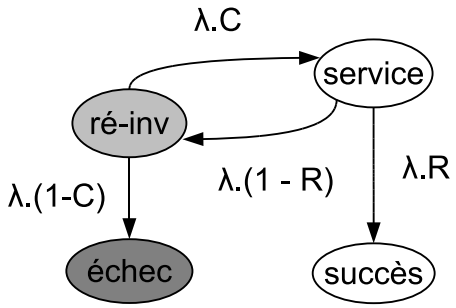


Figure 2.8 – Modélisation de l'échec d'un service avec tentative de ré-invocation

élevée correspond à celui qui est la cause de la dégradation la plus forte de la fiabilité de l'orchestration. Ceci peut être utile pour déterminer le service sur lequel il faut agir pour améliorer la qualité de l'orchestration.

Synthèse : les approches basées sur les modèles probabilistes (réseaux de Petri, chaînes de Markov) ont l'avantage de bénéficier des outils existants, comme SPNP et PRISM, pour la déduction des attributs qualité de l'orchestration globale. Cependant, ces approches supportent seulement la fiabilité et/ou le temps de réponse parmi l'ensemble large des attributs qualité que nous avons recensés (voir section 2.3.1). D'autre part, les approches présentées précédemment ne prennent en compte que les distributions de probabilités de type exponentielle négative, ce qui est restrictif et non pas toujours vrai.

Dans la section suivante, nous présentons les approches basées sur les règles d'agrégation de patrons de *workflow* et nous montrons que certaines de ces limites sont dépassées.

2.4.3 Approches basées sur les règles d'agrégation de patrons de *workflow*

Les travaux [Cardoso et al., 2002, C. Jaeger, 2007, Rosenberg, 2009, Coppolino et al., 2007] ont ciblé l'évaluation des attributs qualité des orchestrations de services en se basant sur des règles d'agrégation. Ces travaux partagent le même principe. L'idée de base consiste à définir des règles d'agrégation pour chaque patron de *workflow* et pour chaque attribut qualité. Dans la plupart des cas, les règles sont définies pour un couple de patrons de *workflow* (dénotés souvent comme « patrons de composition » [C. Jaeger, 2007, Rosenberg, 2009]) sauf pour les patrons « séquence » et « boucle » qui sont considérés individuellement. Un couple de patrons de *workflow* est constitué d'un patron de divergence et d'un patron de convergence (ex. le patron de divergence en parallèle avec le patron de synchronisation). En identifiant les patrons de *workflow* décrivant le modèle de l'orchestration, la procédure consiste à appliquer les règles d'agrégation correspondantes à chaque attribut qualité. Cette étape est réalisée en commençant par les patrons de *workflow* les plus imbriqués dans le modèle de l'orchestration jusqu'à ce qu'il soit totalement parcouru. Les valeurs des différents attributs qualité obtenues à la fin de la procédure correspondent à celles de l'orchestration globale.

Tableau 2.2 – Pertinence des patrons de *workflow* pour l'agrégation des attributs qualité

Patrons de *workflow*	**Pertinence**	**abstraction**
1 Sequence	Oui	Sequence
2 Parallel Split	Oui	AND-Split
3 Synchronization	Oui	AND-Join
4 Exclusive Choice	Oui	XOR-Split
5 Simple Merge	Oui	XOR-Join
6 Multi Choice	Oui	OR-Split
7 Synchronizing Merge	Oui	OR-Join
8 Multi Merge	Oui	AND-Join
9 Discriminator	Oui	m-out-of-n
10 Arbitrary Cycles	Oui	Loop
11 Implicit Termination	Non	–
12 Multiple Instance (MI) without Synchronization	Oui	AND-Split
13 MI with a priori Design Time Knowledge	Oui	représenté par les patrons 2-9
14 MI with a priori Runtime Knowledge	Non	–
15 MI without a priori Runtime Knowledge	Non	–
16 Deferred Choice	Oui	XOR-Split
17 Interleaved Parallel Routing	Oui	Sequence
18 Milestone	Non	–
19 Cancel Activity	Oui	–
20 Cancel Case	Non	–

[C. Jaeger, 2007] a étudié les vingt patrons de *workflow* [van der Aalst et al., 2003] issus du premier bilan de WPI [van der Aalst and ter Hofstede, 1999]. Il en ressort que certains patrons de *workflow* ne sont pas pertinents pour l'agrégation des attributs qualité notamment pendant la phase de conception de l'orchestration. Le tableau 2.2 résume les patrons de *workflow* et leur pertinence pour l'agrégation des attributs qualité. L'idée de base sur laquelle repose l'étude est que seuls les patrons relatifs à la structure logique de l'orchestration en phase de conception sont pertinents ; les patrons liés à la phase d'exécution n'ont pas d'impact sur l'agrégation des attributs qualité pendant la phase de conception comme les patrons « *implicit termination* » et « *cancel case* » [C. Jaeger, 2007]. Les patrons de *workflow* impliquant les instances multiples sont également destinés à l'exécution. Si le nombre d'instances est connu pendant la phase de conception (patrons 12 et 13), ceci est traduit par une divergence parallèle (AND-Split). Dans le cas où le nombre d'instances est déterminé au cours de l'exécution (patrons 13 et 14), l'estimation des attributs qualité ne peut pas être faite. Par conséquent, ces patrons ne sont pas pris en compte pour l'agrégation des attributs qualité. Le patron « *milestone* » décrit la capacité de déclencher une activité externe à un moment donné de l'orchestration. Un tel comportement n'a pas d'incidence sur les attributs qualité de l'orchestration. Par conséquent, ce comportement est jugé comme non pertinent pour l'agrégation des attributs qualité.

La colonne abstraction dans le tableau 2.2 représente la notation utilisée pour chaque patron de *workflow*. Ces abstractions sont utilisées également dans d'autres travaux [Cardoso et al., 2002, Rosenberg, 2009]. L'interprétation de « AND » est que tous les services sont impliqués. Le « XOR » signifie qu'un seul service est impliqué tandis que le « OR » met en jeu plus d'un service mais moins que la totalité des services possibles. Le « *m-out-of-n* » (ou m/n) signifie que parmi les n services mis en parallèle, m services sont requis pour la synchronisation, avec $m < n$.

À partir des vingt patrons de *workflow* [C. Jaeger, 2007] a recensé neuf patrons de composition jugés pertinents pour l'agrégation des attributs qualité (voir figure 2.9). [Rosenberg, 2009] a considéré quatre patrons de composition qui sont la séquence, la boucle, le parallélisme (*AND-AND*) et le choix exclusif (*XOR-XOR*). [Cardoso et al., 2002] ont pris en compte cinq patrons de composition tandis que [Coppolino et al., 2007] ont proposé six patrons de composition pour la fiabilité. Le tableau 2.3 récapitule les patrons de composition identifiés dans [Cardoso et al., 2002, C. Jaeger, 2007, Rosenberg, 2009, Coppolino et al., 2007] ainsi que les attributs qualité pris en compte.

Les règles d'agrégation proposées dans ces travaux présentent quelques différences : [Cardoso et al., 2002, Coppolino et al., 2007] traitent le patron « boucle » avec des probabilités pour chaque branche sortante (voir figure 2.10b). [C. Jaeger, 2007, Rosenberg, 2009] annotent les boucles avec une estimation du nombre d'itérations l (voir figure 2.10a). Le nombre d'itérations peut être estimé en analysant l'historique des exécutions. Le cas le plus simple consiste à prendre la moyenne du nombre d'itérations des exécutions précédentes. D'autre part, [C. Jaeger, 2007] agrège les valeurs minimales et les valeurs maximales des attributs qualité alors que [Cardoso et al., 2002, Rosenberg, 2009, Coppolino et al., 2007]

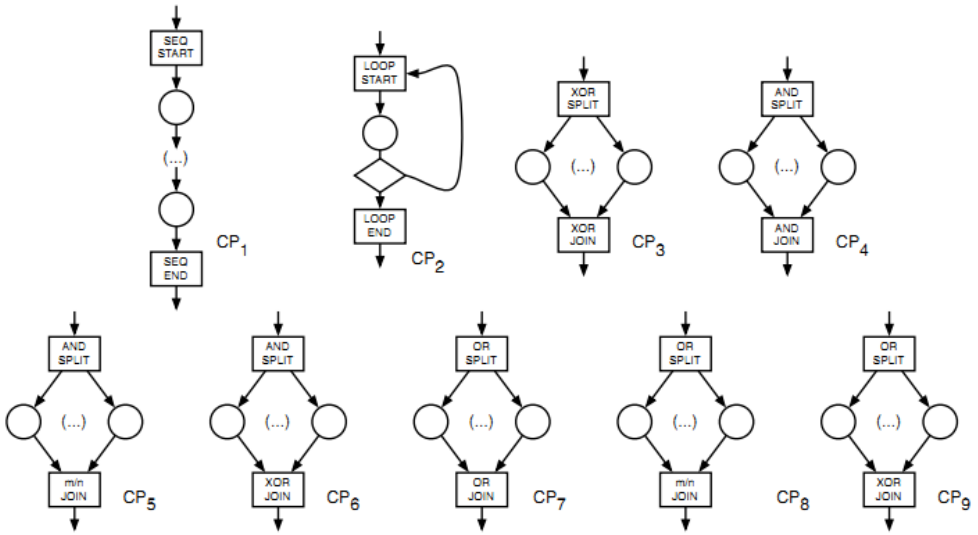


Figure 2.9 – Patrons de composition dans [C. Jaeger, 2007]

agrègent les valeurs moyennes. Le tableau 2.4 présente les règles d'agrégation du temps de réponse qui ont été proposées dans [C. Jaeger, 2007, Rosenberg, 2009, Cardoso et al., 2002].

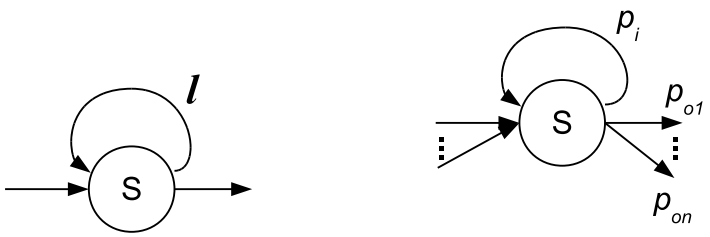


Figure 2.10 – (a) Modélisation des boucles par estimation du nombre d'itérations ; (b) Modélisation des boucles par attribution de probabilités aux branches sortantes

Dans ces règles d'agrégation, n représente le nombre de services impliqués dans le patron de composition, $q_{rt}(s_i)$ est le temps de réponse du service s_i, l est le nombre estimé d'itérations, p_i est la probabilité de ré-exécution pour le patron boucle et la probabilité qu'un service s_i soit choisi pour le patron de choix exclusif, $\mathbb{K}$ dénote l'ensemble de toutes les combinaisons possibles du choix multiple, et $\min_{\substack{1\leq i\leq n \\ k}}(q_{rt}(s_i))$ détermine les k valeurs minimales parmi $(q_{rt}(s_1), \ldots, (q_{rt}(s_n)$.

Pour le patron de composition XOR-XOR, la moyenne pondérée est utilisée dans [Rosenberg, 2009, Cardoso et al., 2002] pour donner une estimation des attributs qualité. [C. Jaeger, 2007] précise que la moyenne pondérée ne représente qu'une minorité d'exécutions possibles. En plus, il considère que le calcul de la moyenne des valeurs moyennes n'est pas utile. Il estime qu'il est plus adéquat de définir les bornes minimales et maximales

Tableau 2.3 – Récapitulation des travaux basés sur les règles de patrons de *workflow*

Travaux de recherche	**Patrons de composition**	**Attribut(s) qualité**
[Cardoso et al., 2002]	séquence, boucle, AND-AND , XOR-XOR, AND-m/n	Temps de réponse, coût, fiabilité, fidélité
[C. Jaeger, 2007]	séquence, boucle, AND-AND, XOR-XOR, AND-XOR, AND-m/n, OR-OR, OR-m/n, OR-XOR	Débit, temps de réponse, coût, disponibilité, réputation, fidélité degré de cryptage
[Rosenberg, 2009]	séquence, boucle, AND-AND, XOR-XOR	Débit, temps de réponse, coût, disponibilité, réputation, degré de cryptage, adaptabilité, précision, robustesse
[Coppolino et al., 2007]	séquence, boucle, *Synchronizing Parallel*, *Multi-Merge Parallel*, *XOR Parallel*, *Discriminator Parallel*	Fiabilité

Tableau 2.4 – Règles d'agrégation du temps de réponse dans la littérature

Patrons de composition	**[C. Jaeger, 2007]**		**[Rosenberg, 2009]**	**[Cardoso et al., 2002]**
	$\mathbf{max(q_{rt})}$	$\mathbf{min(q_{rt})}$		
1- Séquence	$\sum_{i=1}^{n} q_{rt}(s_i)$	$\sum_{i=1}^{n} q_{rt}(s_i)$	$\sum_{i=1}^{n} q_{rt}(s_i)$	$\sum_{i=1}^{n} q_{rt}(s_i)$
2- Boucle	$l.q_{rt}(s)$	$l.q_{rt}(s)$	$l.q_{rt}(s)$	$\frac{q_{rt}(s)}{1-p_i}$
3- XOR-XOR	$\max_{1\leq i\leq n}(q_{rt}(s_i))$	$\min_{1\leq i\leq n}(q_{rt}(s_i))$	$\sum_{i=1}^{n} p_i.q_{rt}(s_i)$	$\sum_{i=1}^{n} p_i.q_{rt}(s_i)$
4- AND-AND	$\max_{1\leq i\leq n}(q_{rt}(s_i))$	$\max_{1\leq i\leq n}(q_{rt}(s_i))$	$\max_{1\leq i\leq n}(q_{rt}(s_i))$	$\max_{1\leq i\leq n}(q_{rt}(s_i))$
5- AND-m/n	$\max_{1\leq i\leq n}(q_{rt}(s_i))$	$\min_{1\leq i\leq n}(q_{rt}(s_i))$	-	$\min_{\substack{1\leq i\leq n \\ k}}(q_{rt}(s_i))$
6- AND-XOR	$\max_{1\leq i\leq n}(q_{rt}(s_i))$	$\min_{1\leq i\leq n}(q_{rt}(s_i))$	-	-
7- OR-OR	$\max_{1\leq i\leq n}(q_{rt}(s_i))$	$\min\left(max(\mathbb{K})\right)$	-	-
8- OR-m/n	$\max_{1\leq i\leq n}(q_{rt}(s_i))$	$\min\left(max(\mathbb{K})\right)$	-	-
9- OR-XOR	$\max_{1\leq i\leq n}(q_{rt}(s_i))$	$\min_{1\leq i\leq n}(q_{rt}(s_i))$	-	-

des attributs qualité (quantitatifs) que de fournir une moyenne de chacun des attributs qualité. Par contre, si les attributs qualité possèdent des distributions de probabilités, la méthode d'agrégation doit les prendre en compte [C. Jaeger, 2007].

D'autre part, il existe plusieurs patrons de convergence en parallèle comme par exemple les patrons *Synchronization* et *Multi-Merge*. Ces patrons n'ont pas tous le même sens. Nous trouvons les patrons de convergence en parallèle *avec synchronisation* (ex. *Synchronization*) et les patrons de convergence en parallèle *sans synchronisation* (ex. *Multi-Merge*). La convergence avec synchronisation signifie que le service qui suit la mise en parallèle, n'est exécuté que lorsque tous les services mis en parallèle ont fini leurs exécutions. Inversement, la convergence sans synchronisation signifie que le service qui suit la mise en parallèle, est exécuté chaque fois qu'un des services de la mise en parallèle termine son exécution. Autrement dit, il y a autant d'instances du service suivant la mise en parallèle que de nombre de services en parallèle. Par conséquent, selon le type de convergence, le service suivant le patron de parallélisme peut être exécuté une seule fois ou n fois (n étant le nombre de services en parallèle). Ceci peut donc avoir un impact sur les attributs qualité qu'il faut prendre en compte dans les règles d'agrégation. [Cardoso et al., 2002,Rosenberg, 2009] prennent en compte seulement les patrons de convergence en parallèle avec synchronisation. [C. Jaeger, 2007] ne différencie pas les deux cas, il représente les deux types de convergence sous la même abstraction AND-join (voir tableau 2.2). [Coppolino et al., 2007] distinguent les deux types de convergence dans les règles d'agrégation présentées. Cependant, ils n'ont traité que l'attribut qualité de fiabilité.

Synthèse : nous avons présenté dans cette section les approches basées sur les règles d'agrégation de patrons de *workflow* pour l'évaluation des attributs qualité des orchestrations. Ces approches montrent quelques avantages par rapport aux approches basées sur les modèles probabilistes. D'une part, ces approches prennent en compte un ensemble plus large d'attributs qualité (voir tableau 2.3). D'autre part, elles sont extensibles : d'autres patrons de *workflow* peuvent être ajoutés et de nouveaux attributs qualité peuvent être intégrés. Cependant, dans la plupart de ces travaux [C. Jaeger, 2007, Rosenberg, 2009, Cardoso et al., 2002], la distinction entre la convergence sans et avec synchronisation n'a pas été étudiée. Bien que [Coppolino et al., 2007] aient adressé ce problème, les attributs qualité pris en compte se limitent seulement à la fiabilité. Nous avons recensé également d'autres différences dans les travaux basés sur les règles de patrons de *workflow*. Ces différences concernent, d'une part, le traitement du patron boucle (voir figure 2.10) et la nature des valeurs agrégées (valeurs minimales, valeurs maximales ou valeurs moyennes) d'autre part. D'autres travaux basés sur les règles de patrons de *workflow* ont adressé l'agrégation des distributions de probabilités des attributs qualité, c'est ce que nous allons voir dans la section suivante.

2.4.4 Agrégation des distributions de probabilités des attributs qualité

Peu de travaux [Menascé et al., 2010, Hwang et al., 2007] ont adressé l'agrégation des distributions de probabilités des attributs qualité. Les travaux de [Menascé et al., 2010] ciblent le domaine de sélection des services. Ils proposent une solution optimale pour sélectionner les services satisfaisant des contraintes globales sur le temps d'exécution et le coût de l'orchestration. Pour vérifier les contraintes globales, ils utilisent les règles d'agrégation des patrons de *workflow* (voir section 2.4.3). Les patrons de *workflow* utilisés ainsi que les règles d'agrégation associées sont similaires à ceux de [Rosenberg, 2009]. La seule différence concerne le nombre d'itérations du patron « boucle »qui peut être assimilé à une variable aléatoire discrète quand il n'est pas fixe. Dans ces travaux, le temps d'exécution est considéré comme une variable aléatoire définie par sa densité de probabilité (*Probability Density Function*) et/ou sa fonction de répartition (*Cumulative Distribution Function*). Les informations sur les variables aléatoires sont supposées fournies par le fournisseur de services ou obtenues par analyse de l'historique des exécutions. La démarche proposée pour la vérification des contraintes globales consiste à calculer l'expression analytique de la variable aléatoire de toute l'orchestration et de comparer sa moyenne (espérance mathématique) avec la contrainte spécifiée. Pour ce faire, des formules mathématiques sont présentées pour calculer :

- l'espérance mathématique du maximum de n variables aléatoires indépendantes,
- la densité de probabilité et la fonction de répartition de la somme de n variables aléatoires indépendantes.

Le calcul direct de la densité de probabilité de la somme de n variables aléatoires *continues* est fastidieux. En effet, la densité de probabilité (p_{1+2}) de la somme de deux variables aléatoires est le produit de convolution de leurs densités de probabilité individuelles p_1 et

p_2 :

$$p_{1+2}(x) = \int_{y=0}^{y=+\infty} p_1(y).p_2(x-y)dy \ .$$

La fonction de répartition (P_{1+2}) correspondante est la primitive de la densité de probabilité résultante :

$$P_{1+2}(x) = \int_{y=0}^{y=x} p_{1+2}(y)dy \ .$$

Pour obtenir la densité de probabilité de la somme de n variables aléatoires, il faut calculer récursivement deux par deux la densité de probabilité des variables aléatoires additionnées. Pour simplifier le calcul, les auteurs utilisent la transformée de Laplace [Grove, 1991]. Ainsi, la transformée de Laplace de la densité de probabilité de la somme de n variables aléatoires $\mathcal{L}\Big[p_{1+2+...+n}(X)\Big]$ est le produit des transformées de Laplace de leurs n densités de probabilité :

$$\mathcal{L}\Big[p_{1+2+...+n}(X)\Big] = \prod_{i=1}^{n} \mathcal{L}\Big[p_i(X)\Big] \ .$$

La densité de probabilité de la somme des n variables aléatoires est obtenue ensuite en utilisant la transformée inverse de Laplace [Grove, 1991]. Cependant, ce calcul est faisable dans le cas où les variables aléatoires suivent *des distributions exponentielles*. Dans le cas contraire, ce calcul reste difficile voire impossible à réaliser au delà de deux variables aléatoires.

[Hwang et al., 2004, Hwang et al., 2007] proposent une approche similaire (basée sur les règles de patrons de *workflow*) pour l'agrégation des attributs qualité probabilistes. Les attributs qualité considérés sont le temps de réponse, la fiabilité, la réputation et le coût. Les auteurs ne font aucune hypothèse sur les distributions de probabilités des attributs qualité. Pour assurer une telle flexibilité (prise en compte de n'importe quelle distribution de probabilités), ils utilisent les lois de probabilité (*Probability Mass Function–PMF*) des attributs qualité définies sur un domaine fini. Autrement dit, chaque attribut qualité est vu comme une variable aléatoire discrète dont la loi de probabilité indique la probabilité d'obtenir une mesure donnée appartenant au domaine. Il est courant d'utiliser les variables aléatoires discrètes pour modéliser la fiabilité, la réputation et le coût. Néanmoins, le temps de réponse peut être assimilé à une variable aléatoire discrète en l'observant avec une fine granularité. Plus la granularité est fine, plus grande est la précision. Une discrétisation possible consiste à partitionner la plage du temps de réponse en un ensemble de sous intervalles. Chaque sous intervalle est représenté par une valeur (ex. le maximum) avec une probabilité. Ainsi, le temps de réponse est modélisé par une variable aléatoire discrète selon une loi de probabilité. Un algorithme est présenté également pour déterminer la densité de probabilité et la fonction de répartition d'une variable aléatoire discrète à partir de sa loi de probabilité. Étant donné les lois de probabilité de tous les services participant dans l'orchestration, le chalenge réside dans la détermination des lois de probabilité résultant des opérations d'agrégation. Pour ce faire, des opérations basiques sur les lois de probabilité ont été introduites. Ces opérations, utilisées dans les règles d'agrégation, sont

l'addition, la multiplication, le maximum, le minimum et le choix exclusif. En appliquant ces opérations d'agrégation sur n lois de probabilité, chacune de dimension k, il s'avère que la dimension du domaine de la variable agrégée explose rapidement ; elle est de l'ordre de k^n. Dans ce contexte, les auteurs ont proposé deux méthodes différentes pour réduire la dimension du domaine de la variable agrégée après chaque opération d'agrégation tout en minimisant « l'erreur d'agrégation ». L'erreur d'agrégation mesure la différence introduite entre la variable aléatoire réduite et l'originale. La méthode retenue s'appelle la méthode *greedy* [Hwang et al., 2004, Hwang et al., 2007]. Elle consiste à fusionner récursivement le couple de points adjacents ayant l'erreur d'agrégation minimale jusqu'à atteindre la dimension souhaitée. Notons que le problème d'explosion du domaine ne se manifeste pas pour l'attribut qualité de fiabilité dont le domaine se restreint aux valeurs 0 et 1. Des expériences ont été conduites pour valider les méthodes de discrétisation et l'approche globale. Les distributions de probabilités des variables aléatoires choisies pour conduire ces expériences sont les distributions normales (définies par leurs moyennes et leurs déviations standards). Le premier type d'expérience, destiné à la validation de la méthode de discrétisation, consiste à comparer la fonction de répartition théorique de la somme de deux variables aléatoires avec celle résultante de la méthode d'agrégation. Les résultats montrent que les fonctions de répartition sont très proches (différence inférieure à 0.01). Le deuxième type d'expérience a pour objectif de vérifier la fonction de répartition de l'orchestration globale obtenue par l'approche d'agrégation. Pour ce faire, une simulation est réalisée dix fois. À chaque simulation, un million d'instances de l'orchestration sont exécutées. Une loi de probabilité du temps de réponse est obtenue à l'issue de chaque simulation et la moyenne des dix lois de probabilité a servi comme une référence pour la loi de probabilité du temps de réponse globale de l'orchestration. La différence constatée entre la loi de probabilité référence et la loi de probabilité obtenue par la méthode d'agrégation (utilisant la méthode *greedy*) est minime (inférieure à 0.001). Par conséquent, l'approche présentée dans ces travaux est assez précise malgré les différentes opérations réalisées sur les variables aléatoires continues (discrétisation et réduction du domaine de la variable aléatoire).

Synthèse : les travaux de [Menascé et al., 2010] ne prennent en compte que le temps de réponse. La méthode présentée pour l'agrégation des distributions de probabilités du temps de réponse n'est réalisable que dans le cas des distributions exponentielles négatives. Nous considérons que cette hypothèse est restrictive, le temps de réponse peut être modélisé avec d'autres types de distributions de probabilités. En outre, nous estimons que dans le cas où les distributions sont exponentielles et que l'attribut qualité considéré est le temps de réponse, il est plus simple d'utiliser les modèles probabilistes (voir section 2.4.2). Quant aux travaux de [Hwang et al., 2004,Hwang et al., 2007], ils sont destinés à la phase de conception de l'orchestration et plus particulièrement pour la sélection des services impliqués dans l'orchestration. Aucun des travaux cités précédemment n'a traité l'agrégation des attributs qualité probabilistes durant la phase d'exécution de l'orchestration.

2.4.5 Gestion de la qualité des orchestrations de services au cours de l'exécution

La plupart des travaux portant sur l'agrégation des attributs qualité, notamment les règles de patrons de *workflow*, concernent la sélection des services [Huang et al., 2009, C. Jaeger, 2007, Rosenberg, 2009, Menascé et al., 2010], pendant la phase de conception, dans le but de vérifier la satisfaction des contraintes globales de l'orchestration. Dans cette section, nous présentons les travaux de [Canfora et al., 2008, Qiao et al., 2009, Ben Halima et al., 2008, Szydlo and Zielinski, 2008, He et al., 2008] qui ont ciblé la phase d'exécution de l'orchestration et plus particulièrement l'adaptation de la qualité de l'orchestration en cas de dégradation.

Dans [Canfora et al., 2008, Canfora et al., 2005b, Canfora et al., 2005a], les auteurs présentent une approche pour déclencher la sélection des services de l'orchestration au cours de l'exécution. Le but est de préserver la satisfaction des contraintes globales (sur l'orchestration) en cas de déviation des valeurs des attributs qualité au cours de l'exécution. En effet, les valeurs des attributs qualité peuvent dévier de leurs valeurs précédemment estimées pendant la phase de conception. De plus, le nombre d'itérations des boucles ainsi que les chemins d'exécution des choix conditionnels dépendent des données à l'exécution et peuvent également engendrer des déviations des valeurs des attributs qualité de l'orchestration. Enfin, dans le cas extrême, certains services peuvent être indisponibles au moment de leurs invocations. L'approche proposée consiste à évaluer les attributs qualité au cours de l'exécution et de déclencher la replanification du reste de l'orchestration en cas de forte déviation ou risque de violation du contrat (SLA). L'approche s'appuie sur les règles d'agrégation de patrons de *workflow* pour évaluer les attributs qualité de l'orchestration. Au cours de l'exécution, les règles d'agrégation et par la suite les valeurs des attributs qualité, sont mises à jour au fur et à mesure que de nouvelles informations sont disponibles : nombre d'itérations réellement exécutées, branches conditionnelles sélectionnées, mesures des attributs qualité, etc. La déviation des valeurs des attributs qualité engendrant le déclenchement de la replanification est représentée par deux seuils. Le premier seuil (global) concerne la valeur de l'attribut qualité portant sur l'orchestration tandis que le deuxième seuil (local) indique le pourcentage maximum de déviation de l'attribut qualité du service invoqué. La replanification consiste à trouver l'ensemble des services de la partie non exécutée de l'orchestration de manière à assurer les mêmes besoins fonctionnels et réduire au maximum la déviation. Ainsi, seule la substitution des services est considérée comme scénario d'adaptation. Le mécanisme de sélection est basé sur une fonction d'utilité globale définie de telle sorte qu'elle maximise les attributs qualité de dimension croissante et minimise ceux qui ont une dimension décroissante (voir section 2.3.1). La fonction d'utilité utilisée dans [Canfora et al., 2008] est définie de la manière suivante :

$$F = \left(\sum_i w_i \cdot q_i \right) (1 - w_d \cdot D) \ . \tag{2.1}$$

La première partie de cette fonction d'utilité correspond à une moyenne pondérée, tandis que la deuxième partie représente un facteur pénalisant le score global lorsqu'il y a des violations de contraintes. En effet, le poids w_d représente l'importance du facteur pénalisant D calculé comme étant la somme de chacune des distances par rapport à la contrainte violée par l'attribut qualité en question. Les valeurs des attributs qualité sont normalisées dans un intervalle [0,1] à l'aide d'équations linéaires [Zeng et al., 2004]. Ainsi, les services sont choisis de sorte qu'ils maximisent la fonction d'utilité et satisfassent les contraintes locales et globales. Cependant, la construction de la fonction d'utilité, notamment l'affectation des poids d'importance relatifs aux attributs qualité et au facteur pénalisant, est à la charge du concepteur (utilisateur). Nous considérons que cette tâche n'est pas facile pour le concepteur si nous voulons une modélisation fidèle de ses préférences sur les attributs qualité.

[Qiao et al., 2009] présentent une architecture pour l'adaptation dynamique des orchestrations de services. L'approche consiste à changer dynamiquement le chemin d'exécution de l'orchestration lorsque cela s'avère nécessaire. Pour ce faire, les auteurs ont introduit un nouveau constructeur dans BPEL appelé « *flexpath* ». Ce constructeur permet de définir des chemins d'exécution multiples. Pendant la phase de conception, l'architecte définit les parties de l'orchestration qui sont susceptibles d'être modifiées au cours de l'exécution. Dans chacune de ces parties, un ensemble de chemins d'exécution alternatifs (*flexpath*) est spécifié. Durant l'exécution, l'adaptation est déclenchée au moment où les valeurs des attributs qualité sont jugées inacceptables. Le mécanisme d'adaptation se charge de trouver le prochain constructeur *flexpath* et choisit le meilleur chemin d'exécution. Dans ces travaux, seuls l'attribut qualité « temps de réponse » et le patron de *workflow* « séquence » sont étudiés. La règle de déclenchement de l'adaptation est définie comme suit :

$$T_m + T_f > T_c * f$$

où $0 < f < 100\%$ représente le facteur de prédiction, T_m, T_f et T_c désignent respectivement le temps de réponse de la partie exécutée, le temps de réponse estimé de la partie restante et le temps de réponse globale contractualisé. Le temps de réponse estimé est calculé sur la base de l'historique d'exécution des services. Les auteurs ont conduit des expériences en variant le facteur de prédiction f. Les résultats montrent qu'en prenant $f = 100\%$, la violation du contrat se produit fréquemment alors qu'avec un facteur de prédiction de 80% les violations sont très rares. Ces travaux présentent quelques limitations. La spécification des chemins d'exécution alternatifs (les stratégies d'adaptation) se fait avant l'exécution. Par conséquent, l'adaptation est planifiée (voir section 2.2.2). De plus, elle n'est pas optimale. En effet, il se peut que les chemins alternatifs définis ne permettent pas de satisfaire les contraintes globales. Inversement, dans certains cas, un chemin d'exécution moins contraignant peut remédier à la dégradation survenue au cours de l'exécution.

[Ben Halima et al., 2008] proposent une architecture prédictive auto-adaptative des services web. L'approche se base sur la supervision et l'analyse des attributs qualité. Les attributs qualité considérés dans ces travaux sont liés au temps, à savoir le temps de réponse, le temps d'exécution et la latence (voir section 2.3.1). La surveillance se fait sur l'évolution des valeurs des attributs qualité plutôt que sur les valeurs elles-mêmes. Les auteurs considèrent que les valeurs des attributs qualité (ex. le temps de réponse) peuvent différer selon le contexte de l'application (bande passante, plateforme, etc) alors que l'évolution des valeurs des attributs qualité dans le même contexte reflètent l'état du service (bon fonctionnement ou défaillance). Ainsi, l'augmentation continue du temps de réponse est considérée comme un signe d'une future défaillance du service. En particulier, la violation successive de N temps de réponse signifie la défaillance du service. Une expression analytique permettant de minorer la valeur de N est donnée en se basant sur l'attribut qualité « précision » et la probabilité de transition de l'état du service :

$$N \geq 1 + \frac{\log\left(\frac{q_{acr}(s,o)}{P[AnyState \rightarrow Violation]}\right)}{\log(P[Violation \rightarrow Violation])}$$

où $q_{acr}(s, o)$ est la valeur de l'attribut qualité précision, $P[AnyState \rightarrow Violation]$ représente la probabilité de transition de n'importe quel état vers l'état de violation et $P[Violation \rightarrow Violation]$ est la probabilité de rester dans l'état de violation. Le seuil de violation du temps de réponse est la somme du temps de réponse moyen du service et d'un délai de tolérance proportionnel à la déviation standard. Lorsque la violation survient successivement N fois, les auteurs présentent des algorithmes pour localiser la dégradation et identifier éventuellement le service responsable. Étant donné que le temps de réponse est la somme du temps d'exécution et de la latence, la localisation de la dégradation consiste à déterminer si la cause provient du réseau ou des services. Dans le cas où la dégradation est causée par les services, il est nécessaire d'identifier le service responsable parmi l'ensemble des services impliqués dans l'orchestration pour que les actions d'adaptation apportées soient efficaces. Les actions d'adaptation considérées sont [Pegoraro et al., 2008] :

- la substitution du service responsable de la dégradation par un autre service équivalent offrant une meilleure qualité ;
- la duplication de service (distribution de la charge entre le service initial et un autre service équivalent).

L'algorithme d'analyse de la source de dégradation ne traite que le cas des services en séquence. Les autres patrons de *workflow* (ex. parallélisme) ne sont pas étudiés. Une dégradation du temps de réponse d'un service A mis en parallèle avec un service B peut être masquée par ce dernier lorsque son temps de réponse est plus élevé.

Dans [Szydlo and Zielinski, 2008], les auteurs proposent une méthode adaptative pour le contrôle de la qualité des orchestrations de services. Il s'agit de changer dynamiquement les contrats (SLA) établis afin de respecter le budget alloué par le client. Cette stratégie d'adaptation est différente de la substitution des services. Elle suppose que les services

impliqués dans l'orchestration sont fournis avec différentes offres de qualité et par conséquent différents coûts. L'approche proposée vise à sélectionner les offres (SLA) adéquates de service afin d'assurer le respect du budget du client. Les contrats (SLA) choisis sont ceux qui correspondent aux meilleures valeurs des attributs qualité selon les préférences du client et tels que le coût total n'excède pas son budget. Pour ce faire, une fonction d'utilité définie comme étant la moyenne pondérée des valeurs des attributs qualité permet de choisir les meilleures offres des services. Les auteurs ont utilisé une méthode d'aide à la décision multi-critère appelée *Analytic Hierarchy Process* (AHP) [Forman and Selly, 2001]. Cette méthode est utilisée pour déterminer les poids d'importance des attributs qualité à partir des préférences exprimées par le client. Cependant, aucune méthode de normalisation des attributs qualité n'est mentionnée dans ces travaux. L'adaptation est réalisée quand il y a une déviation des valeurs des attributs qualité de l'orchestration au cours de l'exécution. Les valeurs des attributs qualité calculées au cours de l'exécution se basent sur les mesures des attributs qualité des services déjà exécutés et des estimations des attributs qualité du reste de l'orchestration. Les estimations sont obtenues par des techniques de régression, telle que la moyenne statistique, appliquées sur l'historique des invocations de services. Cependant, les patrons de *workflow* étudiés dans cette approche sont restreints uniquement au patron de séquence. D'autre part, la moyenne pondérée utilisée dans la fonction d'utilité ne permet pas de modéliser précisément les préférences du client. Dans certaines situations, le client peut présenter des dépendances entre ses préférences sur les attributs qualité [Marichal, 2000, Marichal, 99, Clivillé, 2004, Grabisch and Labreuche, 2005]. Ces dépendances préférentielles peuvent impacter et fausser le résultat de la fonction d'utilité et par suite les décisions qui en découlent. Pour cette raison, nous estimons qu'il est important de prendre en compte les dépendances préférentielles afin de mieux représenter les préférences du client. Plusieurs travaux ont montré que la moyenne pondérée ne permet pas de prendre en compte les dépendances qui peuvent exister entre les préférences du client sur les attributs qualité [Grabisch and Labreuche, 2008, Grabisch and Labreuche, 2005]. À ce jour, aucun des travaux n'a considéré la prise en compte des dépendances préférentielle pour modéliser les préférences du client dans un but de gestion de la qualité des orchestrations de services.

[He et al., 2008] présentent également une méthode basée sur les patrons de *workflow* pour mesurer l'impact de l'adaptation des orchestrations de services. À la différence des autres approches, ils présentent un nouveau concept de « la valeur de l'information changée » (*Value Of Changed information – VOC*) qui représente le profit apporté par le changement de l'orchestration. L'adaptation est réalisée lorsque le profit est supérieur au coût de l'adaptation (dû à la négociation avec les fournisseurs de services, changement des services, etc). Les stratégies d'adaptation considérées concernent la re-négociation d'un nouveau contrat (SLA) avec le même fournisseur de service ou la substitution du service défaillant et potentiellement d'autres services dans le reste de l'orchestration. En effet, lorsqu'un service ne satisfait pas les contraintes des attributs qualité spécifiées dans le

contrat (SLA), il ne s'agit pas de choisir le service avec les meilleures valeurs des attributs qualité. Les auteurs identifient trois facteurs à prendre en compte pour une adaptation optimale :

- un compromis entre la VOC (profit) et le coût des services candidats doit être estimé (le profit doit être supérieur au coût),
- l'impact des services adaptés sur les autres services doit être estimé. L'objectif est de décider s'ils ont besoin d'être adaptés également afin de satisfaire les contraintes globales de l'orchestration,
- le coût d'adaptation des autres services doit être pris en compte.

Les auteurs ont étudié les patrons de *workflow* [van der Aalst et al., 2003] et en ont dégagé sept patrons de composition qui sont : *séquence, Parallel split + Synchronizing, Parallel split + Multi-Merge, Parallel split + Discriminator, Multi-choice + Synchronizing Merge, Multi-choice + Multi-Merge et Multi-choice + Discriminator.* Ils ont proposé ensuite des expressions analytiques pour ces patrons de composition permettant d'estimer chacun des facteurs présentés ci-dessus et par la suite la VOC globale de l'adaptation du service en question. Dans les expressions proposées, les auteurs mentionnent l'utilisation des règles d'agrégation de patrons de *workflow* [Hwang et al., 2007] pour évaluer les attributs qualité de l'orchestration et une méthode d'aide à la décision multi-critères pour vérifier la satisfaction des contraintes globales. Les détails sur la méthode d'aide à la décision multi-critères utilisée dans ces travaux ne sont pas fournis. D'autre part, l'adaptation est supposée avoir lieu si le profit estimé est supérieur au coût associé à l'adaptation ; les coûts liés à l'adaptation et à l'éventuelle négociation avec les fournisseurs sont spécifiques au domaine d'application et ne sont pas pris en compte dans l'approche. De plus, devant la prolifération du nombre de services équivalents susceptibles de remplacer le service, l'application de l'approche à tous les services nécessite un calcul intensif. Par ailleurs, nous pensons que les stratégies d'adaptation considérées dans ces travaux ne sont pas adéquates pour l'adaptation dynamique de l'orchestration en cours d'exécution. En effet, les attributs qualité utilisés pour la démonstration de l'approche sont le temps de réponse et le coût. Dans ce cas, la violation du contrat est détectée après la fin de l'exécution du service en question. Par conséquent, la substitution du service à l'origine de la dégradation permet de résoudre le problème pour l'exécution de la prochaine instance de l'orchestration et non pas celui de l'instance en cours d'exécution.

2.5 Synthèse

Dans le contexte des architectures orientées services, la notion de qualité est devenue incontournable dans le développement et la supervision des applications. Nous avons vu dans la section 2.3.1 qu'il existe plusieurs attributs qualité dans le domaine des AOS. Ces attributs qualité ont plus ou moins d'importance pour les clients (utilisateurs finaux, experts, etc) selon le domaine d'application et les exigences des clients. Ces dernières sont exprimées par le biais des contrats (SLA) (voir section 2.3.2) spécifiant ainsi le niveau

de qualité garanti par les fournisseurs des services de telle manière que les contraintes des clients soient satisfaites. Lors de la conception de l'orchestration, les clients peuvent spécifier deux types de contraintes :

- des contraintes locales portant sur les attributs qualité des services participant à l'orchestration ;
- des contraintes globales portant sur les attributs qualité de l'orchestration globale.

Afin de vérifier les contraintes globales, il convient d'agréger les valeurs des attributs qualité des services impliqués dans l'orchestration en prenant en compte sa structure logique. Nous avons présenté plusieurs travaux ciblant cet objectif (voir section 2.4). Les travaux basés sur les approches d'agrégation probabilistes ont l'avantage de s'appuyer sur des méthodes formelles (chaînes de Markov, réseaux de Petri) qui possèdent des outils de vérification existants et largement validés par la communauté scientifique. Néanmoins, ces travaux ne supportent qu'un nombre limité d'attributs qualité à savoir le temps de réponse et la fiabilité. En outre, les distributions de probabilités de ces attributs qualité sont supposées être exponentielles, ce qui n'est pas forcément le cas (voir section 2.4.2). Les travaux d'agrégation basés sur les règles de patrons de *workflow* apportent plus de flexibilité. En effet, le nombre d'attributs qualité supportés est plus large et n'est pas limité : de nouveaux attributs qualité ainsi que d'autres patrons de *workflow* peuvent être intégrés. Nous avons vu dans la section 2.4.4 qu'il est possible d'agréger les distributions de probabilités des attributs qualité sans fixer une hypothèse sur leurs distributions de probabilités [Hwang et al., 2007]. Cependant, ces travaux ciblent la phase de conception de l'orchestration. Dans le même contexte, [C. Jaeger, 2007] a proposé des règles d'agrégation de valeurs minimales et maximales tandis que [Cardoso et al., 2002, Rosenberg, 2009, Coppolino et al., 2007] ont considéré les valeurs moyennes des attributs qualité. Nous considérons que ces travaux non seulement ne ciblent que la phase de conception, mais qu'ils ne fournissent pas une estimation précise de la qualité des orchestrations. Bien que les règles proposées dans [Cardoso et al., 2002, Rosenberg, 2009, Coppolino et al., 2007] puissent être appliquées à la fin de l'exécution, l'agrégation des attributs qualité au cours de l'exécution reste peu étudiée aujourd'hui.

Dans cette thèse, nous nous focalisons sur la gestion de la qualité des orchestrations de services en cours d'exécution. Par conséquent, il est nécessaire d'évaluer les attributs qualité tout au long de l'exécution. [Canfora et al., 2008, Szydlo and Zielinski, 2008, Qiao et al., 2009] sont les seuls travaux, à notre connaissance, qui ont adressé cette problématique. Par contre, à un instant t de l'exécution, ils considèrent les mesures des attributs qualité des services exécutés et les moyennes statistiques des attributs qualité de la partie restante de l'orchestration. Nous estimons que ceci représente une simplification qui risque fort de ne pas donner une bonne estimation de la qualité de l'orchestration. Ainsi, nous pensons que cette problématique doit être approfondie.

Par ailleurs, nous avons vu qu'il y a quelques différences dans les règles d'agrégation de patrons de *workflow* (voir section 2.4.3). La première différence concerne le traitement du patron *boucle* (voir figure 2.10). [Rosenberg, 2009, C. Jaeger, 2007] annotent le patron

boucle avec une constante qui dénote le nombre d'itérations tandis que [Cardoso et al., 2002, Coppolino et al., 2007] annotent chaque branche sortante du patron avec des probabilités. Il convient donc de choisir le modèle adéquat du patron boucle selon que l'on connait a priori le nombre d'itérations ou non. La deuxième différence concerne le nombre de patrons de composition pris en compte (voir tableau 2.3). [Rosenberg, 2009] a considéré quatre patrons de composition, [Cardoso et al., 2002] ont pris en compte cinq patrons de composition tandis que [Coppolino et al., 2007] en ont proposé six. [C. Jaeger, 2007] a étudié les vingt patrons de *workflow* issus du premier bilan de WPI [van der Aalst et al., 2003] et a identifié le plus grand nombre de patrons de composition qui est au nombre de neuf (voir figure 2.9). Certains de ces travaux [C. Jaeger, 2007, Rosenberg, 2009, Cardoso et al., 2002] n'ont pas distingué entre les patrons de *workflow* de convergence avec synchronisation et ceux qui sont sans synchronisation. Bien que [Coppolino et al., 2007] différencient les deux cas, les attributs qualité pris en compte se limitent à la fiabilité.

Nous estimons par ailleurs qu'un système de supervision doit prendre en considération les préférences du client vis-à-vis des attributs qualité. Les préférences du client dépendent du domaine métier de l'application et des exigences du client. À titre d'exemple, une légère dégradation du temps de réponse des processus de longue durée peut être tolérée. Les préférences du client peuvent être traduites en partie par des poids d'importance attribués à chacun des attributs qualité considérés. Peu de travaux ont considéré les préférences du client [Szydlo and Zielinski, 2008, He et al., 2008]. [He et al., 2008] ne précisent pas la méthode utilisée pour prendre en compte les préférences du client tandis que [Szydlo and Zielinski, 2008] ont utilisé la méthode AHP pour déterminer les poids d'importance des attributs qualité. Cette méthode est basée sur l'opérateur d'agrégation qui est la moyenne pondérée ; un opérateur qui ne prend pas en compte les dépendances préférentielles du client [Clivillé, 2004, Berrah et al., 2008]. Dans ce sens, nous estimons qu'un système de supervision doit se doter d'un modèle de préférences précis (c'est-à-dire qui prend en compte les éventuelles dépendances préférentielles du client). Ceci est dans le but ne pas avoir des déclenchements de l'adaptation et des adaptations elles-mêmes qui sont *(i)* parfois inutiles lorsque la satisfaction globale du client est assurée bien qu'une contrainte sur un attribut qualité soit violée ou *(ii)* insuffisantes dans le sens où les adaptations améliorent mais ne rétablissent pas la satisfaction du client ou encore *(iii)* surabondantes lorsque les adaptations apportent plus que le nécessaire en terme de satisfaction du client. À notre connaissance, cette problématique n'a pas encore été étudiée.

Le tableau 2.5 présente un bilan des travaux réalisés sur la composition des orchestrations de services [Cardoso et al., 2002, C. Jaeger, 2007, Rosenberg, 2009, Hwang et al., 2007, Sato and Trivedi, 2007, Gallotti et al., 2008, Zhong and Qi, 2006] et la gestion de leur qualité [Canfora et al., 2008, Szydlo and Zielinski, 2008, He et al., 2008, Qiao et al., 2009, Ben Halima et al., 2008]. Le signe « + » signifie que le critère est pris en compte, tandis que « - » indique le contraire. Le signe « +/- » signifie que le critère est pris en compte partiellement. Nous allons nous intéresser particulièrement à la gestion dynamique des attributs qualité non déterministes, liés à l'exécution et portant sur l'orchestration

Tableau 2.5 – Synthèse de l'état de l'art

	Composition de services	**Gestion dynamique de la qualité**
Attributs qualité probabilistes	+	-
Agrégation des attributs qualité	+	+
Préférences du client	-	+/-
Dépendances préférentielles	-	-

globale. Nous considérons qu'il est important de prendre en compte les préférences des clients et leurs dépendances préférentielles vis-à-vis des attributs qualité dans la gestion de la qualité des orchestrations au cours de l'exécution. Le tableau montre qu'aucun des travaux actuels ne considère les dépendances préférentielles dans la gestion de la qualité des orchestrations en cours d'exécution. En outre, la détection des dégradations dans ces travaux est basée uniquement sur la vérification des contraintes ; aucun de ces travaux ne s'est basé sur la surveillance de la satisfaction du point de vue de l'utilisateur. D'autre part, la prise en compte des distributions de probabilités des attributs qualité au cours de l'exécution de l'orchestration, quant à elle, n'a pas été étudiée. Ce sont principalement ces problématiques que nous allons étudier dans la suite de ce document.

Chapitre 3

Agrégation des attributs qualité dans un cadre de supervision des orchestrations de services

3.1 Introduction

La supervision des architectures orientées services, plus précisément des orchestrations de services, est un sujet de recherche récent qui a suscité de plus en plus d'intérêt durant les cinq dernières années. En particulier, la supervision de la qualité des orchestrations en cours d'exécution est encore peu étudiée aujourd'hui. Aucun des travaux que nous avons recensés dans ce domaine [Ben Halima et al., 2008,Canfora et al., 2008,Szydlo and Zielinski, 2008, Qiao et al., 2009] n'a pris en compte les préférences des clients (consommateurs, utilisateurs, etc) dans la surveillance de la qualité des orchestrations de services. En outre, les dépendances préférentielles que peut présenter le client, n'ont jamais été considérées dans les approches de gestion dynamique de la qualité des orchestrations de services.

Par ailleurs, le suivi de la qualité des orchestrations nécessite une approche d'agrégation qui prend en compte les différentes phases du cycle de vie de des orchestrations de services à savoir pendant la conception, durant l'exécution et à la fin de l'exécution. Là encore, les travaux existants présentent certaines limitations. En effet, la plupart des travaux traitant l'agrégation des attributs qualité se sont focalisés sur la phase de conception des orchestrations [Zhong and Qi, 2006, Gallotti et al., 2008, Sato and Trivedi, 2007, C. Jaeger, 2007, Cardoso et al., 2002, Rosenberg, 2009, Coppolino et al., 2007]. Bien que les travaux de [Cardoso et al., 2002, Rosenberg, 2009, Coppolino et al., 2007] puissent être appliqués à la fin de l'exécution, l'agrégation au cours de l'exécution est peu étudiée aujourd'hui. Seuls [Canfora et al., 2008,Szydlo and Zielinski, 2008,Qiao et al., 2009] ont traité l'agrégation des attributs qualité durant l'exécution. Cependant, les grandeurs utilisées comme estimations des attributs qualité sont des moyennes statistiques, ce qui est non précis notamment en ce qui concerne le temps de réponse. D'autre part, nous avons vu que les travaux basés sur les règles d'agrégation de patrons de *workflow* proposent ou prennent en compte un

nombre variable de patrons de composition. En outre, ils présentent quelques différences et certaines limitations, notamment en ce qui concerne le traitement de certains patrons de *workflow*, la prise en compte des différents type de convergence, etc (voir section 2.5).

Nos travaux s'orientent donc vers ces problématiques et ces limitations afin d'améliorer les travaux existants et résoudre les problématiques qui n'ont pas encore été traitées. Dans la section 3.2, nous introduisons notre approche globale de supervision et nous fixons les objectifs des travaux de cette thèse. Ensuite, nous proposons une approche d'agrégation permettant d'agréger les attributs qualité selon la période de l'évaluation ; c'est-à-dire pendant la phase de conception, durant l'exécution et à la fin de l'exécution. Pour des raisons de présentation, nous commençons par exposer l'agrégation des attributs qualité à la fin de l'exécution (section 3.3). Puis, nous présentons l'approche d'agrégation pendant la phase de conception (section 3.4). Enfin, nous proposons une approche d'agrégation durant l'exécution des orchestrations (section 3.5).

3.2 Introduction à l'approche de supervision

3.2.1 Objectifs de la thèse

Les limitations et les problématiques que nous avons soulevées précédemment servent de base pour l'élaboration de notre contribution. Ainsi, nos travaux ont pour objectifs de :

- proposer une approche d'agrégation des attributs qualité :
  - pendant la phase de conception des orchestrations ;
  - durant l'exécution des orchestrations ;
  - à la fin de l'exécution des orchestrations.
- étudier les patrons de *workflow* issus de la dernière révision de WPI [Russell et al., 2006a], en particulier les combinaisons possibles de patrons (en adéquation avec les patrons de composition recensés dans la littérature) et en proposer les règles d'agrégation indéfinies ;
- construire un modèle de préférences des clients qui prend en compte les dépendances préférentielles entre les attributs qualité ;
- évaluer le degré de satisfaction des orchestrations tout en considérant les exigences des clients et leurs modèles de préférences ;
- gérer l'adaptation dynamique des orchestrations en proposant :
  - des stratégies de surveillance (*monitoring*) ;
  - des politiques de déclenchement de l'adaptation dynamique selon la stratégie de surveillance ;
  - des scénarios d'évolution.

Ces objectifs, que nous nous sommes fixés, entrent dans le cadre d'une approche globale de supervision des orchestrations de services. Cette approche globale a les particularités de :

- être indépendante de n'importe quel langage d'orchestration ou de description de *workflow* (voir section 3.3.4) ;
- supporter un ensemble large d'attributs qualité (voir section 3.3.3) ;
- considérer n'importe quelle distribution de probabilités pour décrire les attributs qualité (voir section 3.4).

Dans la section suivante, nous introduisons le cadre général de notre approche de supervision avant d'entrer dans ses détails.

3.2.2 Cadre général de l'approche de supervision

La figure 3.1 décrit le cadre général dans lequel s'inscrit notre approche de supervision. Lors de la phase de conception, le client commence par recenser les fonctionnalités dont il a besoin et décrire leur structure logique permettant d'accomplir ses besoins métiers. Ceci est appelé *le processus abstrait*, c'est une orchestration de fonctionnalités ; les services concrets n'étant pas encore sélectionnés. Après avoir spécifié le processus abstrait, le client définit également *ses exigences* en terme de qualité requise. Ceci revient à spécifier les contraintes locales et globales sur les attributs qualité qui doivent être satisfaites lors du choix des services (concrets). Le client peut également exprimer ses préférences sur les attributs qualité pour être prises en compte dans la sélection des services. Dans le contexte de notre approche, l'expression des préférences du client nous sert à construire un *modèle de préférences* qui va être exploité dans l'évaluation de la qualité globale de l'orchestration. Nous aborderons plus en détails ces notions d'exigences et de modèle de préférences dans le chapitre suivant (voir sections 4.2 et 4.3). *La sélection des services* consiste, par suite, à chercher dans l'*annuaire des services*, les services *(i)* assurant les fonctionnalités spécifiées dans le processus abstrait et *(ii)* satisfaisant les exigences du client et (éventuellement) ses préférences sur les attributs qualité. Le résultat de la sélection se traduit par *un modèle d'orchestration de services* (concrets) prêt à être instancié et exécuté. À ce stade, le client établit potentiellement *des contrats (SLA)* (voir section 2.3.2) avec *les fournisseurs des services* spécifiant le niveau de qualité garanti et les conditions d'utilisation de ces services. L'étape de sélection est en dehors du périmètre de cette thèse, nous orientons le lecteur vers [Sathya et al., 2011] pour une revue de la littérature des techniques de sélection des services pour les orchestrations. Cependant, nous présentons dans le chapitre 5 (voir section 5.2) une application de cette démarche dans un domaine particulier qui est le pilotage d'ateliers de production.

À l'exécution, le modèle de l'orchestration est instancié et exécuté par le biais *d'un moteur d'exécution* dédié au langage d'orchestration utilisé. À titre d'exemple, il existe plusieurs moteurs d'exécution pour le langage d'orchestration WS-BPEL parmi lesquels nous pouvons citer ActiveBpel, Apache ODE et Oracle Bpel Process Manager [Lapadula et al., 2008]. Enfin la tâche cruciale réside dans la supervision des instances d'orchestrations en cours d'exécution. Nous pouvons décomposer cette tâche en trois fonctions : *l'acquisition*, *l'agrégation* et *la gestion de l'adaptation*. L'acquisition consiste en *la mesure des attributs qualité*. Il existe plusieurs techniques dans la littérature permettant de

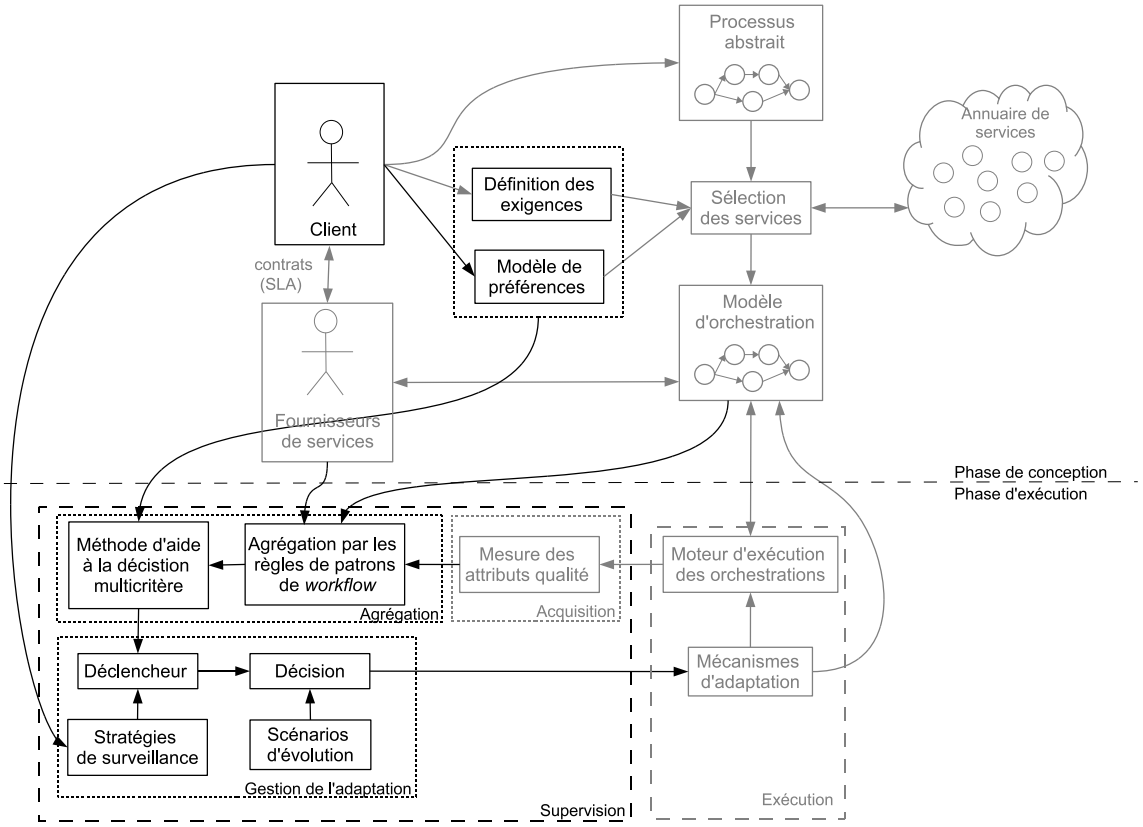


Figure 3.1 – Contexte de nos travaux de recherche

mesurer les attributs qualité parmi lesquels nous citons :

- l'analyse des fichiers log de l'exécution des services [Zo et al., 2007]. Cette technique est souvent utilisée pour mesurer la fiabilité des services.
- l'utilisation des mécanismes d'interception des messages qui transitent entre le client et les fournisseurs des services (*proxy*) [Ben Halima et al., 2008]. Cette technique permet de mesurer notamment la performance (temps de réponse, latence, débit, etc).
- l'analyse du trafic réseau (paquets TCP) [Rosenberg, 2009] pour mesurer les attributs qualité liés au temps à savoir le temps de réponse, la latence et le temps d'exécution.
- l'instrumentation du moteur d'exécution de l'orchestration moyennant la programmation orientée aspect (*Aspect Oriented Programming*) [Zhang et al., 2008] afin de collecter des informations sur les services et en analyser des propriétés incluant quelques attributs qualité comme la sécurité, la fiabilité et la performance.
- l'utilisation des sondes (*probing*) [Wang et al., 2009] exploitée pour mesurer particulièrement la disponibilité des services.

Nos principales contributions s'inscrivent dans les fonctions d'agrégation et de gestion de l'adaptation du système de supervision (voir figure 3.1). La fonction d'agrégation consiste en deux phases : une phase *d'agrégation par les règles de patrons de workflow* et une phase d'agrégation utilisant *une méthode d'aide à la décision multi-critères*. Nous introduisons

le principe de ces deux phases dans la section suivante. Retenons juste que cette approche d'agrégation (par les deux phases) fournit le degré de satisfaction de la qualité globale de l'orchestration par rapport aux exigences définies par le client. C'est sur cette information de haut niveau que va se baser *la gestion de l'adaptation* de l'orchestration. En effet, la gestion de l'adaptation est composée par plusieurs fonctions. La première fonction, assurée par *le déclencheur*, consiste à surveiller le degré de satisfaction, résultant de l'approche d'agrégation. Selon *la stratégie de surveillance* choisie par le client, le déclencheur vérifie les conditions de déclenchement de l'adaptation dynamique. Les stratégies de surveillance et les conditions de déclenchement correspondantes font le sujet du chapitre suivant (voir section 4.5). Lorsque l'adaptation dynamique est déclenchée, un ou plusieurs *scénarios d'évolution* seront mis en place afin d'améliorer la qualité globale de l'orchestration. Les scénarios d'évolution adéquats sont à sélectionner par la fonction de *décision* selon le degré de dégradation et les attributs qualité concernés. Cette fonction de décision fera le sujet de nos futurs travaux. Nous proposons dans le chapitre suivant (voir section 4.5) plusieurs scénarios d'évolution susceptibles d'améliorer la qualité des orchestrations. Ces scénarios, lorsqu'ils sont choisis par la partie *décision*, seront intégrés dans l'instance en cours d'exécution (éventuellement dans le modèle d'orchestration aussi pour les futures instances) par le biais *des mécanismes d'adaptation* propres au moteur d'exécution des orchestrations. Cela forme une boucle de rétroaction comme c'est le cas dans la théorie de contrôle des systèmes automatisés [Kesraoui, 2010] (voir figure 3.1).

Le contexte générale de nos travaux étant défini, nous procédons maintenant à la présentation du principe de l'approche d'agrégation qui correspond à la fonction d'analyse du système de supervision (voir figure 3.1).

3.2.3 Principe de l'approche d'agrégation

Étant donné les services impliqués dans l'orchestration, chaque service possède des valeurs de plusieurs attributs qualité. Supposons que nous avons p attributs qualité différents $q_1, ..., q_p$. Nous définissons le niveau de qualité d'un service s_i, noté $QoSL(s_i)$, comme un vecteur de valeurs d'attributs qualité :

$$QoSL(s_i) = (q_1(s_i), ..., q_p(s_i))$$

où $q_1(s_i), ..., q_p(s_i)$ représentent les valeurs des p attributs qualité du service s_i et déterminent le niveau de qualité de ce service.

Le principe de l'approche d'agrégation est illustré dans la figure 3.2. Elle comprend deux grandes phases. La première phase exploite une méthode permettant de calculer les valeurs des attributs qualité de l'orchestration globale (voir section 2.4). Notre choix a porté sur une méthode basée sur les règles d'agrégation de patrons de *workflow*. Ce choix est justifié du fait que cette méthode est flexible et extensible. Elle est extensible car le nombre de patrons de *workflow* ainsi que le nombre des attributs qualité supportés ne sont pas fixés. La flexibilité réside dans le choix de représentation des attributs qualité :

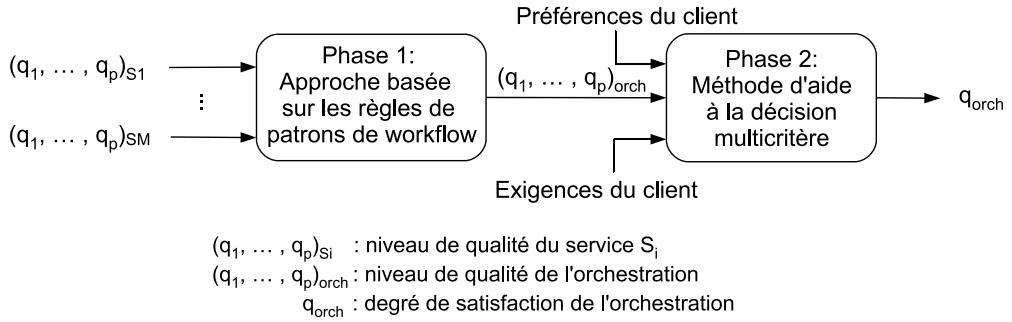


Figure 3.2 – Principe de l'approche d'agrégation

la méthode est applicable aussi bien avec des valeurs déterministes (fixes) qu'avec des variables aléatoires d'attributs qualité. Ceci est très pertinent pour notre approche. En effet, nous pouvons distinguer trois périodes d'évaluation de la qualité des orchestrations :

- pendant la phase de conception de l'orchestration : dans cette période, les informations que nous possédons sur les attributs qualité sont des estimations. La plupart des travaux de recherche [Canfora et al., 2008,Rosenberg, 2009] prennent comme estimations des valeurs d'attributs qualité, les moyennes statistiques. Dans notre approche, nous considérons que les distributions de probabilités des attributs qualité sont plus adéquates pour donner des estimations plus précises des valeurs d'attributs qualité, en particulier pour le temps de réponse. Par conséquent, l'agrégation par les patrons de *workflow* considère les attributs qualité comme des variables aléatoires définies par des distributions de probabilités ;
- durant l'exécution de l'orchestration : dans cette période, nous disposons d'une part des mesures de chacun des attributs qualité de la partie de l'orchestration qui a été exécutée, et des estimations de la partie restante d'autre part. Bien entendu, les estimations ici sont les distributions de probabilités des attributs qualité (de nature stochastique) ;
- après la fin de l'exécution de l'orchestration : dans cette période, tous les services étant exécutés, nous avons toutes les mesures d'attributs qualité pour tous les services. Par conséquent, l'agrégation par les patrons de *workflow* considère les valeurs déterministes (fixes).

Nous utilisons donc la méthode d'agrégation basée sur les règles de patrons de *workflow* pour déterminer la valeur ou la distribution de probabilités de chaque attribut qualité $q_i(orch)$ $(1 \leq i \leq p)$ de l'orchestration globale à partir des valeurs ou des distributions de probabilités d'attributs qualité $q_i(s_j)$ $(1 \leq j \leq M)$ des M services impliqués dans l'orchestration. Ceci nous permet d'obtenir un seul vecteur de valeurs et/ou de distributions de probabilités d'attributs qualité $QoSL(orch)$ de l'orchestration globale à partir des M vecteurs $QoSL(s_1), ..., QoSL(s_M)$ des services qui la composent (voir figure 3.2).

Cependant, ce vecteur (de valeurs ou de distributions de probabilités) d'attributs qualité de l'orchestration présente une difficulté d'interprétation, notamment lorsque l'on

considère plusieurs attributs qualité. Prenons l'exemple de trois attributs qualité : fiabilité, disponibilité et temps de réponse. Admettons que les valeurs limites (contraintes globales) de ces attributs qualité, acceptées par le client, soient respectivement *0.8, 0.7* et *300s* pour toute l'orchestration. Supposons, maintenant, que les valeurs perçues des attributs qualité de l'orchestration (obtenues par l'approche d'agrégation basée sur les règles de patrons de *workflow*) soient respectivement *0.85, 0.68* et *260s*. Les questions qui se posent sont :

- que peut-on dire de l'écart entre chacune de ces valeurs perçues et les valeurs limites correspondantes (contraintes globales) ? Autrement dit, à quel point les valeurs perçues satisfont les contraintes posées par le client ou inversement à quel point elles violent ces contraintes ?
- est-ce que le client est globalement satisfait de la qualité perçue de l'orchestration par rapport à son investissement (c'est-à-dire sa sélection de services et le budget alloué pour obtenir cette qualité), notamment lorsqu'il y a un mélange de bonnes et de mauvaises valeurs ? Est-ce que les bonnes valeurs de certains attributs qualité compensent ou surpassent les mauvaises valeurs des autres attributs qualité ou vice-versa ?

En se basant uniquement sur le vecteur d'attributs qualité de l'orchestration, il ne serait pas possible de répondre aux questions soulevées précédemment. Tout dépend des préférences du client, de l'importance accordée à chaque attribut qualité et du processus métier en question. Par conséquent, nous considérons que l'appréciation de la qualité globale de l'orchestration (c'est-à-dire le vecteur d'attributs qualité) est spécifique à chaque client et au contexte d'utilisation. Il est donc important d'intégrer le comportement et les préférences du client dans l'analyse de la qualité globale de l'orchestration. D'autre part, l'appréciation du client vis-à-vis de la qualité globale de l'orchestration peut se montrer une tâche très complexe du point de vue cognitif, notamment lorsque l'on considère plusieurs attributs qualité.

Pour résoudre cette problématique, nous utilisons dans la deuxième phase de notre approche d'agrégation (voir figure 3.2) une méthode d'aide à la décision multi-critères. Cette méthode permet de déterminer les satisfactions de chacune des valeurs dans le vecteur d'attributs qualité de l'orchestration et de les agréger ensuite à l'aide d'une fonction d'utilité globale F telle que :

$$F\Big(QoSL(orch)\Big) = q_{orch}$$

où q_{orch} représente le degré de satisfaction de l'orchestration dans le cas d'agrégation de valeurs déterministes. Dans le cas où l'agrégation se fait sur des distributions de probabilités, nous obtenons la distribution de probabilités du degré de satisfaction de l'orchestration. Cette deuxième phase de l'approche d'agrégation est présentée en détails dans le chapitre suivant (voir section 4.2).

Nous nous focalisons, dans ce chapitre, sur la première phase de l'approche d'agréga-

tion, plus précisément sur l'agrégation par les règles de patrons de *workflow* durant les différentes phases du cycle de vie de l'orchestration.

3.3 Agrégation des valeurs déterministes

L'agrégation des valeurs déterministes concerne la fin de l'exécution où toutes les mesures des attributs qualité se rapportant à tous les services impliqués dans l'orchestration sont disponibles (par la fonction d'acquisition du système de supervision, voir figure 3.1). Pour cela, nous allons :

- étudier les quarante trois patrons de *workflow* issus du dernier bilan de WPI [Russell et al., 2006b] (section 3.3.1) ;
- identifier et compléter la liste des patrons de composition ; couples de patrons de *workflow* (section 3.3.2) ;
- présenter les règles d'agrégation pour les patrons de composition qui existent dans la littérature et en définir des nouvelles pour les autres patrons, et ce pour les attributs qualité que nous allons choisir (section 3.3.3) ;
- enfin, présenter la démarche d'agrégation moyennant les règles d'agrégation et les patrons de composition (section 3.3.4).

3.3.1 Étude des patrons de *workflow*

Les travaux qui ont étudié et/ou utilisé les patrons de composition (voir section 2.4) se sont basés sur le premier bilan du WPI ; c'est-à-dire sur vingt patrons de *workflow* [van der Aalst et al., 2003]. Afin de couvrir un spectre plus large des modèles d'orchestrations, nous allons considérer dans cette thèse le deuxième bilan du WPI qui englobe quarante trois patrons de *workflow* [Russell et al., 2006b]. Ces derniers représentent généralement un raffinement des vingt premiers patrons de *workflow*, notamment en ce qui concerne les patrons de divergence (*Split*) et de convergence (*Join*). À titre d'exemple, le patron *Discriminator* du premier bilan est divisé en six patrons dans la version révisée [Russell et al., 2006b], à savoir *Structured Discriminator*, *Blocking Discriminator*, *Cancelling Discriminator*, *Structured Partial Join*, *Blocking Partial Join* et *Cancelling Partial Join*. Ainsi, la première motivation qui nous a conduits à mener cette étude est la description des patrons de composition existants à l'aide des quarante trois patrons de *workflow*.

D'autre part, en analysant les patrons de composition identifiés dans la littérature [C. Jaeger, 2007, Rosenberg, 2009, Cardoso et al., 2002, Coppolino et al., 2007, He et al., 2008], nous constatons une variation du nombre de patrons de composition pris en compte dans ces travaux. Sachant que les travaux mentionnés précédemment n'ont pas traité les mêmes attributs qualité, ceci implique que certains attributs qualité ne possèdent pas encore des règles d'agrégation pour tous les patrons de composition que nous avons recensés. Ceci nous a poussé à tirer profit de ces travaux et regrouper les différents patrons de composition recensés. L'objectif est de proposer ensuite les règles d'agrégation qui manquent pour les attributs qualité que nous choisissons dans la suite (voir section 3.3.3).

La démarche que nous avons suivie dans cette étude consiste, dans un premier temps, à identifier les patrons de divergence et de convergence parmi les quarante trois patrons de *workflow*. Le résultat est présenté dans le tableau 3.1. À l'exception des patrons de boucle et de séquence, les patrons de composition dans les travaux de [C. Jaeger, 2007,Rosenberg, 2009, Cardoso et al., 2002, He et al., 2008] sont composés d'un patron de divergence, par exemple le patron *XOR-Split*, et d'un patron de convergence comme le patron *XOR-Join*, sauf dans [Coppolino et al., 2007] où un patron de composition peut être composé d'un ou plusieurs patrons de divergence et d'un ou plusieurs patrons de convergence. Notons que nous allons exprimer les patrons de *workflow* impliqués dans les patrons de composition à l'aide de leurs appellations d'origine présentées dans [Russell et al., 2006b] (pour préserver leur sémantique) et non pas par les notations *AND, OR, XOR*, etc. Ensuite, l'idée est de prendre chaque patron de divergence et de convergence, impliqués dans les patrons de composition recensés dans la littérature (voir tableau 2.3), et de chercher les patrons de *workflow* qui leur correspondent parmi les quarante trois patrons ; c'est-à-dire parmi les patrons de *workflow* que nous avons identifiés dans le tableau 3.1. Par exemple, dans le patron de composition *XOR-Split + XOR-Join*, le *XOR-Split* peut correspondre aux patrons de *workflow Exclusive Choice* ou *Deferred Choice* tandis que le *XOR-Join* peut correspondre aux patrons de *workflow Simple Merge* ou *Synchronizing Merge*, etc (voir figure 3.3). Notons également qu'un patron de composition peut représenter plusieurs combinaisons possibles de patrons de *workflow* comme c'est le cas de l'exemple précédent, où le patron de composition « *XOR-Split + XOR-Join* » peut représenter plusieurs combinaisons (voir figure 3.3) parmi lesquelles : *Exclusive Choice / Simple Merge* ou *Deferred Choice / Simple Merge*, etc.

En suivant cette démarche, nous présentons dans la section suivante notre sélection de patrons de composition *(i)* exprimés à l'aide des appellations originales du WPI et *(ii)* regroupant toutes les combinaisons possibles de patrons de *workflow*.

3.3.2 Identification des patrons de composition

Dans la suite de ce document, nous adoptons les notations suivantes :

- PC : patron de composition ;
- un PC est composé d'un ou plusieurs patrons de divergence séparés par des barres obliques « / » et (« + ») d'un ou plusieurs patrons de convergence séparés également par des barres obliques. À tire d'exemple *PC : patron de convergence 1 / patron de convergence 2 + patron de divergence 1 / patron de divergence 2* se lit : le patron de composition regroupe le patron de divergence 1 ou le patron de divergence 2 combiné avec le patron de convergence 1 ou le patron de convergence 2.

En examinant les patrons de composition dans les travaux de [C. Jaeger, 2007, Rosenberg, 2009, Cardoso et al., 2002, Coppolino et al., 2007, He et al., 2008], nous avons décelé onze patrons de composition [Fakhfakh et al., 2012] :

Tableau 3.1 – Patrons de*workflow* de divergence et de convergence

Patrons de divergence	**Patrons de convergence**
1- Parallel Split	6- Synchronization
2- Exclusive choice	7- Simple Merge
3- Multi-Choice	8- Structured Synchronization Merge
4- Deferred Choice	9- Multi-Merge
5- Thread Split	10- Structured Discriminator
	11- Blocking Discriminator
	12- Cancelling Discriminator
	13- Structural Partial Join
	14- Blocking partial Join
	15- Cancelling Partial Join
	16- Generalised AND-Join
	17- Local Synchronizing Merge
	18- General Synchronizing Merge
	19- Thread Merge

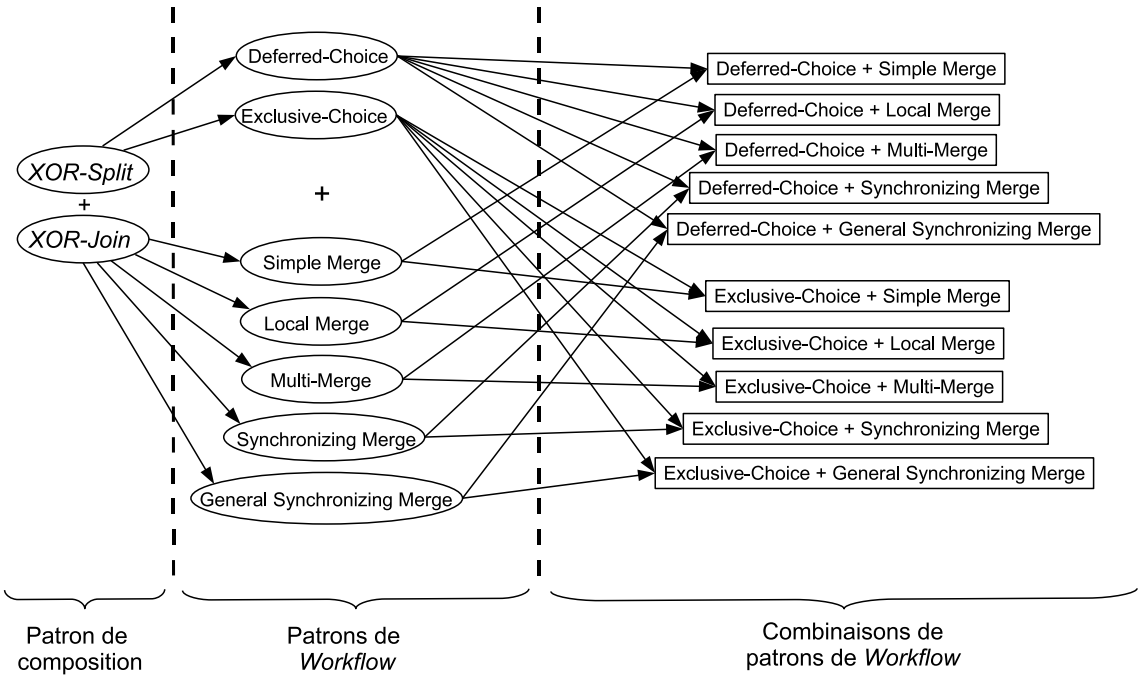


Figure 3.3 – Exemple d'un patron de composition et sa déclinaison en patrons de *workflow* et combinaisons de patrons de *workflow*

PC1 : *Sequence / Interleaved Parallel Routing / Interleaved Routing* : ce patron décrit l'exécution d'un ensemble de services, dans un ordre donné (*Sequence*), dans un ordre partiel (*Interleaved Parallel Routing*) ou dans n'importe quel ordre (*Interleaved Routing*) (voir figure 3.4). Cependant, deux services ne peuvent pas être en exécution en même temps.

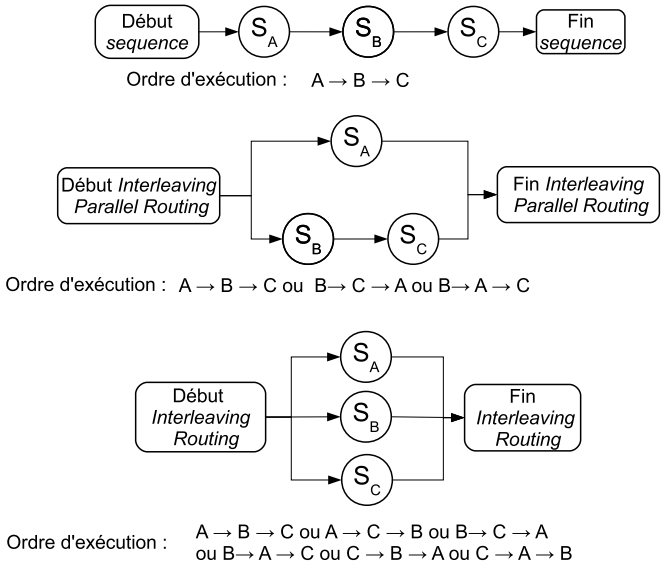


Figure 3.4 – Les patrons de *workflow* appartenant au patron de composition PC1

PC2 : *Structured Loop* : ce patron décrit la répétition de l'exécution d'un service. Contrairement aux travaux de [C. Jaeger, 2007,Rosenberg, 2009,Cardoso et al., 2002], nous notons l le nombre d'itérations de la boucle lorsqu'il est connu (cas déterministe), c'est-à-dire soit à la fin de l'exécution, soit lorsque le nombre d'itérations est fixe et est imposé par la logique métier du processus. Dans le cas contraire, nous considérons le nombre d'itérations comme une variable aléatoire discrète L définie par son domaine $Dom(L) = \{l_1, \ldots, l_c\}$ et sa loi de probabilité P_L, avec l_1 et l_c correspondent respectivement au nombre d'itérations minimum et maximum [Hwang et al., 2007].

PC3 : *Exclusive Choice / Deferred Choice + Simple Merge / Synchronizing Merge / Multi-Merge / Local Merge / General Synchronizing Merge* : ce patron est souvent dénoté par *XOR-XOR* dans [C. Jaeger, 2007,Rosenberg, 2009,Cardoso et al., 2002]. Il décrit le choix de l'exécution d'un service parmi plusieurs services. Le choix du service exécuté dépend des données du processus (*Exclusif Choice*) ou des données externes au processus (*Deferred Choice*). Pendant la phase de conception des orchestrations de services, nous annotons ce patron de composition avec des probabilités p_i, $1 \leq i \leq n$ (n est le nombre de services en parallèle), représentant la probabilité qu'un service i soit choisi.

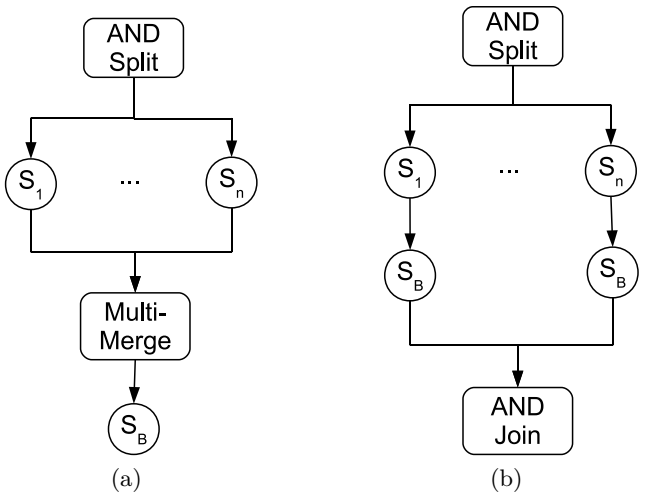


Figure 3.5 – (a) Patron de composition PC5 ; (b) Implémentation alternative du patron de composition PC5

PC4 : *Parallel Split + Synchronization / Structured Synchronization Merge / Generalised AND-Join / Local Synchronizing Merge / General Synchronizing Merge, Thread Split + Thread Merge* : ce patron est souvent dénoté *AND-AND* dans [C. Jaeger, 2007, Rosenberg, 2009, Cardoso et al., 2002]. Dans ce patron, tous les services en parallèle commencent l'exécution au même instant. L'exécution du service qui succède ce patron de composition dans le modèle de l'orchestration ne peut être réalisée que lorsque tous les services, en parallèle, finissent leur exécution. C'est le cas de la convergence avec synchronisation.

PC5 : *Parallel Split + Multi-Merge* : ce patron représente le cas de convergence sans synchronisation. Nous l'avons recensé dans les travaux de [Coppolino et al., 2007, He et al., 2008]. Dans ce patron, tous les services en parallèle commencent l'exécution simultanément. Cependant, le service s_B succédant ce patron (voir figure 3.5a), est exécuté chaque fois qu'un service de la mise en parallèle, finit son exécution. Une implémentation alternative de ce patron de composition consiste à mettre en séquence le service s_B, qui succède le patron de composition, avec chacun des services $s_1, \ldots, s_n$ en parallèle (voir figure 3.5b).

PC6 : *Parallel Split + Structural Partial Join / Blocking Partial Join / Cancelling Partial Join* : parmi n services exécutés simultanément, un nombre m ($m < n$) de services sont requis pour la synchronisation. Par conséquent, le service qui succède à ce patron est exécuté dès que m services ont fini leur exécution. La figure 3.6 montre un exemple du patron PC6 où deux parmi les trois en parallèle sont requis pour la synchronisation. [Cardoso et al., 2002] décrit ce patron de composition comme étant un patron de tolérance aux fautes. Dans ce cas, les services, mis en parallèle, sont équivalents, c'est-à-dire qu'ils assurent la même fonction mais de manière différente. Ceci est utile pour les

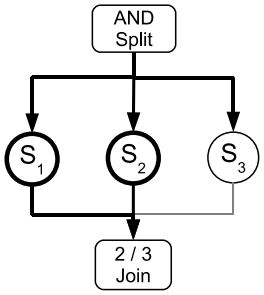


Figure 3.6 – Exemple du patron de composition PC6

processus critiques afin de vérifier les résultats (réponses des services) et assurer la continuité du processus. Ce patron est dénoté *AND-m out of n* dans [C. Jaeger, 2007] (voir figure 2.9). Le nombre de combinaisons possibles (lors de la convergence) de m services parmi n est C_n^m [1]. Prenons l'exemple de trois services s_1, s_2 et s_3, et le cas où deux services sont requis pour la convergence (*AND-2/3*) (voir figure 3.6). L'ensemble des combinaisons possibles, noté $\mathbb{K}$, est $\mathbb{K} = \{(s_1, s_2), (s_1, s_3), (s_2, s_3)\}$.

PC7 : *Parallel Split + Structured Discriminator / Blocking Discriminator / Cancelling Discriminator* : parmi n services exécutés en parallèle, le premier service qui finit son exécution active l'exécution du service succédant le patron, les autres services sont ignorés. Ce patron est dénoté *AND-XOR* dans [C. Jaeger, 2007].

PC8 : *Multi-Choice + Synchronizing Merge / Structured Synchronizing Merge / Generalised AND-Join / Local Synchronizing Merge / General Synchronizing Merge* : ce patron est dénoté *OR-OR* dans [C. Jaeger, 2007]. Parmi n services en parallèle, un nombre k ($1 \leq k \leq n$) de services est choisi, selon une condition donnée, pour être exécutés simultanément. La convergence des services, qui ont été choisis, est effectuée avec synchronisation. Ce patron ressemble au patron PC4 sauf que seuls les services qui ont été choisis lors de la divergence, sont requis pour la synchronisation. Il existe $(2^n - 1)$ choix de combinaisons de services possibles. Notons $\mathbb{K}$ l'ensemble des choix possibles. Le cardinal de $\mathbb{K}$, noté $|\mathbb{K}|$, est égal à $(2^n - 1)$. Notons également par $\mathscr{S} = (s_1, ..., s_k)$ un élément appartenant à $\mathbb{K}$.

L'exécution de ce patron de composition consiste donc à choisir un élément $\mathscr{S}$ de l'ensemble $\mathbb{K}$ qui va être exécuté. Par la suite, nous notons p_i ($1 \leq i \leq 2^n - 1$), la probabilité qu'un élément $\mathscr{S}_i$ de l'ensemble $\mathbb{K}$ soit choisi.

PC9 : *Multi-Choice + Multi-Merge* : ce patron diffère du patron précédent par le type de convergence. La convergence des services qui ont été choisis lors de la divergence pour s'exécuter simultanément, est effectuée sans synchronisation. Ce patron ressemble

1. Rappelons que dans l'analyse combinatoire, le nombre de combinaisons de m objets parmi n et sans remise est $C_n^m = \frac{n!}{m!(n-m)!}$ avec $1 \leq m \leq n$

également au patron PC5 sauf qu'un nombre k ($1 \leq k \leq n$) de service est choisi à la divergence au lieu des n services en parallèle.

PC10 : *Multi-Choice + Structural Partial Join / Blocking Partial Join / Cancelling Partial Join :* ce patron est représenté par *OR-m out of n* dans [C. Jaeger, 2007]. Parmi n services en parallèle, un nombre k de services sont choisis au moment de la divergence pour s'exécuter simultanément ($m \leq k \leq n$). Parmi les k services lancés, un sous ensemble de m services est requis pour la synchronisation. Pour ce patron de composition, le nombre de combinaisons possibles pour la divergence est noté N_1 tandis que le nombre de combinaisons possibles pour la convergence est noté N_2 :

$$N_1 = \sum_{k=m}^{n} C_n^k \qquad\qquad N_2 = \sum_{j=m}^{k} C_k^j$$

PC11 : *Multi-Choice + Structured Discriminator / Blocking Discriminator / Cancelling Discriminator :* ce patron est dénoté *OR-XOR* dans [C. Jaeger, 2007]. Parmi n services en parallèle, k ($1 \leq k \leq n$) services sont choisis pendant l'exécution. Le premier service qui finit son exécution, déclenche l'exécution du service succédant ce patron de composition, les autres services sont ignorés. Ce patron constitue un cas particulier du patron précédent où $m = 1$. Il est aussi semblable au patron de composition PC7 où l'on considère une combinaison de k au lieu de n services à la divergence.

En plus des patrons de convergence et de divergence que nous avons considérés dans notre étude (voir tableau 3.1), il existe d'autres patrons de *workflow* dans [Russell et al., 2006b] que nous n'avons pas mentionnés. Ces patrons incluent principalement les patrons d'instances multiples et les patrons d'annulation. Les patrons concernant les instances multiples peuvent être traduits en fonction d'autres patrons de convergence et de divergence présentés dans cette section. À titre d'exemple, le patron d'instances multiples sans synchronisation (*Multiple Instances without Synchronization*) peut être assimilé au patron PC5 en termes de QdS. D'autre part, les patrons incluant l'annulation et la terminaison n'ont pas d'impact sur l'agrégation des attributs qualité (voir section 2.4.3). Enfin, le patron de *workflow* de cycles arbitraires (*Arbitrary Cycles*) représente les boucles qui possèdent des branches d'entrée et de sortie arbitraires. Ce patron entre dans la catégorie des modèles d'orchestration non structurés (voir section 3.3.4). Par conséquent, il n'est pas pris en compte dans la liste de patrons de composition que nous avons présentée.

Les onze patrons de composition ainsi identifiés n'ont jamais été pris en compte dans aucun des travaux, ce qui implique qu'il existe des attributs qualité qui ne possèdent pas des règles d'agrégation relatives à certains de ces patrons de composition.

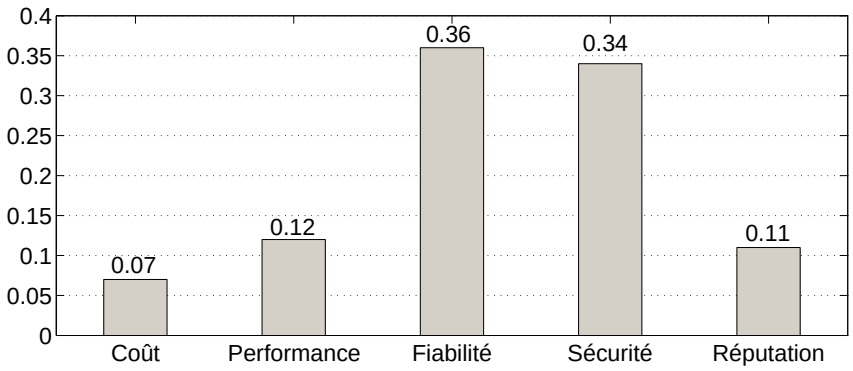


Figure 3.7 – Importance des attributs qualité pour la sélection des services [Zo et al., 2007]

3.3.3 Définition des règles d'agrégation

Les règles d'agrégation sont définies par patron de composition et par attribut qualité. Il convient donc de choisir un ensemble d'attributs qualité afin d'en étudier les règles d'agrégation qui correspondent aux patrons de composition identifiés dans la section précédente. [Zo et al., 2007] ont conduit une enquête auprès de plusieurs architectes des technologies de l'information pour déterminer l'importance des attributs qualité dans la sélection des services. Les résultats (voir figure 3.7) montrent que la fiabilité et la sécurité possèdent les poids d'importance les plus élevés, suivies de la performance, la réputation et le coût. Nous avons déjà écarté les attributs qualité liés à la sécurité du fait de leur nature statique (voir section 2.3.1). Le coût des services est également déterministe (fixe) et ne change pas d'une instance à une autre (voir tableau 2.1). La réputation, quant à elle, a plus d'importance au moment de la sélection des services, c'est-à-dire pendant la phase de conception. De ce fait, ces attributs qualité (sécurité, coût et réputation) sont moins pertinents pour l'illustration de notre approche de supervision au cours de l'exécution (à cause de leur nature statique). Nous estimons que la fiabilité (q_{rel}), la disponibilité (q_{av}) et le temps de réponse (q_{rt}) sont des attributs qualité non déterministes et sont par conséquent pertinents à surveiller et à garantir tout au long de l'exécution des orchestrations.

Ce choix fait sur les attributs qualité n'exclut pas que d'autres attributs qualité puissent être pris en compte par notre approche. En effet, à partir du moment où les attributs qualité possèdent des règles d'agrégation (voir tableau 2.3), ils peuvent être pris en compte par notre approche.

Avant de procéder à la définition des règles d'agrégation pour les trois attributs qualité que nous avons choisis, nous allons présenter un bilan des règles d'agrégation qui existent dans la littérature pour ces trois attributs qualité (voir tableau 3.2). Rappelons que dans cette section, nous traitons l'agrégation par les règles de patrons de *workflow* à la fin de l'exécution. En d'autres termes, nous étudions l'agrégation des valeurs déterministes (fixes) des attributs qualité. Par conséquent, nous considérons uniquement les règles d'agrégation des valeurs déterministes dans ce bilan. Dans le tableau 3.2, le signe « $\surd$ » signifie que l'attribut qualité en question possède une règle d'agrégation dans les travaux cités, le signe

Tableau 3.2 – Bilan des règles d'agrégation de la littérature pour le temps de réponse, la fiabilité et la disponibilité

Patrons de composition	**Temps de réponse**	**Fiabilité**	**Disponibilité**
PC1- Séquence	√ [Rosenberg, 2009, Cardoso et al., 2002]	√ [Cardoso et al., 2002, Coppolino et al., 2007]	√ [Rosenberg, 2009]
PC2- Boucle	√ [Rosenberg, 2009]	-	√ [Rosenberg, 2009]
PC3- XOR-XOR	√ [Rosenberg, 2009, Cardoso et al., 2002]	√ [Cardoso et al., 2002, Coppolino et al., 2007]	√ [Rosenberg, 2009]
PC4- AND-AND	√ [Rosenberg, 2009, Cardoso et al., 2002]	√ [Cardoso et al., 2002, Coppolino et al., 2007]	√ [Rosenberg, 2009]
PC5- AND-AND (sans synch.)	-	√ [Coppolino et al., 2007]	-
PC6- AND-m/n	≈ [Cardoso et al., 2002]	√ [Cardoso et al., 2002, Coppolino et al., 2007]	-
PC7- AND-XOR	-	√ [Coppolino et al., 2007]	-
PC8- OR-OR	-	√ [Coppolino et al., 2007]	-
PC9- OR-OR (sans synch.)	-	√ [Coppolino et al., 2007]	-
PC10- OR-m/n	-	√ [Coppolino et al., 2007] [a]	-
PC11- OR-XOR	-	-	-

a. Bien que [Coppolino et al., 2007] ont proposé six patrons, appelés patrons de fiabilité (voir tableau 2.3, page 35), ces patrons couvrent dix de nos patrons de composition identifiés.

« - » exprime qu'il n'existe pas de règle d'agrégation pour l'attribut qualité concerné et le patron de composition correspondant, et le signe « ≈ » signifie que nous estimons que la règle d'agrégation présentée dans les travaux cités n'est pas correcte (voir ci-après). Nous pouvons constater à travers le tableau 3.2 que chacun des attributs qualité manque de règles d'agrégation pour certains patrons de composition. Dans la suite de cette section, nous allons présenter les règles d'agrégation existantes pour ces attributs qualité et définir de nouvelles règles d'agrégation pour les autres patrons de composition.

Temps de réponse

Les règles d'agrégation pour les onze patrons de composition qui concernent le temps de réponse sont données dans le tableau 3.3. Pour les patrons de séquence (PC1) et de boucle

(PC2), le temps de réponse est déterminé par la somme des temps de réponse de chaque invocation. Le temps de réponse du patron PC3 (*XOR-XOR*) est une moyenne, pondérée par les probabilités p_i de sélection d'un service parmi les n services en parallèle. À l'exécution, la probabilité p_i du service i qui est choisi est égale à 1 et les autres probabilités sont nulles. Cependant, pendant la phase de conception, les probabilités p_i sont déterminées en analysant l'historique d'exécution. Dans le cas de la première exécution, elles sont toutes égales à $\frac{1}{n}$. Pour la divergence en parallèle (*Parallel Split*), différentes règles d'agrégation découlent du type de convergence considéré. Dans le patron PC4 (*AND-AND*), la convergence est avec synchronisation. Par suite, le service succédant au patron n'est exécuté que lorsque tous les services finissent leur exécution. Par conséquent, le temps de réponse du patron PC4 est le maximum des temps de réponse des services en parallèle. Inversement, pour le patron PC7 (*AND-XOR*), le premier service qui finit son exécution déclenche l'exécution du service succédant le patron de composition. Par conséquent, le minimum des temps de réponse des services détermine le temps de réponse global du patron, c'est pourquoi nous définissons la règle d'agrégation suivante pour le patron PC7 :

$$q_{rt}(PC7) = min\left(q_{rt}(s_1), ..., q_{rt}(s_n)\right) \quad . \tag{3.1}$$

Le patron PC5 peut être considéré comme le patron PC4 où une réplication du service s_B, qui succède le patron PC4, est ajoutée en séquence avec chacun des services de la mise en parallèle (voir figure 3.5). Ainsi, nous pouvons déterminer le temps de réponse global de ce patron en prenant le maximum des temps de réponse des services en parallèle ajoutés au temps de réponse du service s_B. Ceci est traduit par la règle d'agrégation suivante :

$$q_{rt}(PC5) = max\left(q_{rt}(s_1) + q_{rt}(s_B), ..., q_{rt}(s_n) + q_{rt}(s_B)\right) \quad . \tag{3.2}$$

Le patron PC6 (*AND-m/n*) implique la convergence avec synchronisation de m services parmi n. Dans ce cas, le nombre de combinaisons de services possibles pour la convergence est C_n^m. Soit $\mathbb{K}$ l'ensemble des combinaisons de services possibles et $q_{rt}(\mathscr{S})$ le temps de réponse d'une combinaison de services $\mathscr{S} = (s_1, ..., s_m)$, avec $q_{rt}(\mathscr{S}) = max\left(q_{rt}(s_1), ..., q_{rt}(s_m)\right)$. Par suite, nous pouvons calculer le temps de réponse du patron PC6, comme une moyenne pondérée des temps de réponse $q_{rt}(\mathscr{S})$ ($\mathscr{S} \in \mathbb{K}$) des combinaisons de services possibles :

$$q_{rt}(PC6) = \sum_{i=1}^{C_n^m} p_i . q_{rt}(\mathscr{S}_i) \quad . \tag{3.3}$$

Les coefficients de pondération p_i représentent les probabilités d'exécution des combinaisons de services $\mathscr{S}_i$. À l'exécution, la probabilité p_i de la combinaison de services $\mathscr{S}_k$ exécutée est égale à 1 et les autres probabilités sont nulles. Par contre, pendant la phase de conception, nous pouvons déterminer les probabilités p_i par analyse de l'historique des exécutions dans le cas où nous en disposons. Dans le cas contraire, nous supposons

Tableau 3.3 – Règles d'agrégation du temps de réponse

Patrons de composition	**Règles d'agrégation**
PC1- Séquence	$\sum_{i=1}^{n} q_{rt}(s_i)$
PC2- Boucle	$l.q_{rt}(s_i)$
PC3- XOR-XOR	$\sum_{i=1}^{n} p_i.q_{rt}(s_i)$
PC4- AND-AND	$max\left(q_{rt}(s_1), ..., q_{rt}(s_n)\right)$
PC5- AND-AND (sans synch)	$max\left(q_{rt}(s_1) + q_{rt}(s_B), ..., q_{rt}(s_n) + q_{rt}(s_B)\right)$
PC6- AND-m/n	$\sum_{i=1}^{C_n^m} p_i.q_{rt}(\mathscr{S}_i)$
PC7- AND-XOR	$min\left(q_{rt}(s_1), ..., q_{rt}(s_n)\right)$
PC8- OR-OR	$\sum_{i=1}^{2^n-1} p_i.max\left(q_{rt}(\mathscr{S}_i)\right)$
PC9- OR-OR (sans synch)	$\sum_{i=1}^{2^n-1} p_i.max\left(q_{rt}(\mathscr{S}_i) + q_{rt}(s_B)\right)$
PC10- OR-m/n	$\sum_{i=1}^{N_1} p_i^1 \left(\sum_{j=1}^{N_2} p_j^2 \max(q_{rt}(\mathscr{S}_i^j))\right)$
PC11- OR-XOR	$\sum_{i=1}^{2^n-1} p_i \min(q_{rt}(\mathscr{S}_i))$

l'équiprobabilité entre les combinaisons de services, c'est-à-dire $p_i = \frac{1}{C_n^m}$.

[Cardoso et al., 2002] ont proposé la règle d'agrégation suivante pour le patron PC6 :

$$\min_{m}(q_{rt}(s_1), \ldots, q_{rt}(s_n)) \ . \tag{3.4}$$

Cette dernière règle fournit les m temps de réponse minimaux parmi les n temps de réponse minimaux des services impliqués dans le patron de composition. En d'autre termes, le résultat de cette règle d'agrégation est un ensemble de valeurs. Nous considérons que ceci n'est pas cohérent avec le principe des règles d'agrégation, dans le cas déterministe (qui est également le cas de ces travaux), qui consiste à fournir une seule valeur de l'attribut qualité pour le patron de composition.

Exemple : prenons cinq service $s_1, ..., s_5$ qui ont respectivement les temps de réponse (en secondes) suivants : 1, 2, 6, 9, 12. L'ensemble $\mathbb{K}$ des combinaisons possibles pour un patron *AND-3/5* est :

$$\mathbb{K} = \{(s_1, s_2, s_3), (s_1, s_2, s_4), (s_1, s_2, s_5), (s_1, s_3, s_4), (s_1, s_3, s_5), (s_1, s_4, s_5), (s_2, s_3, s_4), (s_2, s_3, s_5), (s_2, s_4, s_5), (s_3, s_4, s_5)\}$$

On a cardinal de $\mathbb{K}$: $|\mathbb{K}| = C_5^3 = 10$. Le temps de réponse de chacune des combinaisons de service $\mathscr{S}_i$ appartenant à l'ensemble $\mathbb{K}$ est respectivement $\{6, 9, 12, 9, 12, 12, 9, 12, 12, 12\}$. Si nous considérons le cas où l'on ne dispose pas d'un historique d'exécution, nous avons la probabilité qu'une combinaison de services $\mathscr{S}_i$ soit choisie, égale à $p_i = \frac{1}{C_5^3} = \frac{1}{10}$. L'estimation du temps de réponse global de ce patron de composition pendant la phase

de conception [2] est :

$$q_{rt}(s_1, ..., s_5) = \frac{1}{10} \times (6 + 9 + 12 + 9 + 12 + 12 + 9 + 12 + 12 + 12) = 10.5s$$

Par contre, à l'exécution, une seule probabilité p_i est égale à 1 et les autres sont nulles. Par conséquent, le temps de réponse réel à la fin de l'exécution du patron de composition est, soit 6, soit 9, soit 12, selon la combinaison qui est choisie.

Les mêmes types de convergence utilisés dans les patrons du parallélisme (PC4, PC5, PC6 et PC7) sont considérés pour les patrons de composition de choix multiples (PC8, PC9, PC10 et PC11). La différence entre les deux est que dans les patrons du parallélisme, la divergence concerne tous les services mis en parallèle alors que pour les patrons de choix multiples, un sous ensemble de k services est sélectionné pour l'exécution en parallèle. Par conséquent, le principe des règles d'agrégation reste le même sauf qu'il faut considérer les probabilités que les services soient sélectionnés à la divergence. Pour le patron de composition PC8 (OR-OR), il existe $(2^n - 1)$ combinaisons de choix possibles. Pour une combinaison donnée, le temps de réponse correspondant est le maximum des temps de réponse des services impliqués dans cette combinaison. Par suite, le temps de réponse global du patron de composition peut être estimé par la moyenne des temps de réponse de chaque combinaison possible, pondérée par les probabilités p_i qu'une combinaison i soit choisie. Compte tenu de cela, nous définissons la règle d'agrégation suivante pour le patron PC8 :

$$q_{rt}(PC8) = \sum_{i=1}^{2^n-1} p_i.max\left(q_{rt}(\mathscr{S}_i)\right) \quad . \tag{3.5}$$

À l'exécution, la probabilité p_i est égale à 1 pour la combinaison de services i choisie sinon, pendant la phase de conception, les probabilités p_i sont estimées en analysant l'historique des exécutions. Dans le cas où l'orchestration est exécutée pour la première fois, nous considérons l'équiprobabilité entre les combinaisons possibles, c'est-à-dire $p_i = \frac{1}{2^n-1}$.

Pour le patron PC9, le principe est le même sauf qu'il faut considérer le temps de réponse du service s_B qui succède le patron. Ainsi, nous obtenons la règle d'agrégation suivante :

$$q_{rt}(PC9) = \sum_{i=1}^{2^n-1} p_i.max\left(q_{rt}(\mathscr{S}_i) + q_{rt}(s_B)\right) \quad . \tag{3.6}$$

Pour le patron PC10, nous avons N_1 (respectivement N_2) combinaisons possibles à la divergence (respectivement à la convergence) (voir section 3.3.2). En notant p_i^1 la probabilité qu'une combinaison de service $|\mathscr{S}_i|$ soit choisie lors de la divergence, et p_j^2 la probabilité qu'une combinaison de service $|\mathscr{S}_i^j|$ soit choisie lors de la convergence, avec $\mathscr{S}_i^j$ est une sous-combinaison de $\mathscr{S}_i$, nous pouvons déterminer la règle d'agrégation suivante pour le

2. Notons que l'objectif de cet exemple est d'illustrer l'application de la règle d'agrégation que nous avons proposée. En effet, pendant la phase de conception, nous considérons les attributs qualité comme des variables aléatoires dans notre approche.

patron PC10 :

$$q_{rt}(PC10) = \sum_{i=1}^{N_1} p_i^1 \left(\sum_{j=1}^{N_2} p_j^2 \max(q_{rt}(\mathscr{S}_i^j)) \right) \ . \tag{3.7}$$

Les probabilités p_k^1 et p_k^2 peuvent être estimées si nous disposons d'un historique d'exécutions (voir section 2.4.3). Dans le cas contraire, en particulier dans le cas de la première exécution de l'orchestration, nous considérons l'équiprobabilité de sélection des combinaisons possibles. Dans ce cas, les probabilités p_k^1 et p_k^2 sont égaux à $\frac{1}{N_1}$ et $\frac{1}{N_2}$ respectivement. À la fin de l'exécution, la règle d'agrégation précédente se réduit au maximum des temps de réponse se rapportant aux m services impliqués dans la convergence.

Le patron de composition PC11 est semblable au patron PC7 sauf qu'il faut considérer une combinaison de services $\mathscr{S}_i$ parmi les $(2^n - 1)$ au lieu des n services mis en parallèle. En conséquence, la règle d'agrégation du patron de composition PC11 est :

$$q_{rt}(PC11) = \sum_{i=1}^{2^n-1} p_i \min(q_{rt}(\mathscr{S}_i)) \ . \tag{3.8}$$

Fiabilité et disponibilité

Le tableau 3.4 présente les règles d'agrégation pour la fiabilité et la disponibilité. Ces dernières sont des grandeurs probabilistes (voir section 2.3.1). En conséquence, les valeurs agrégées sont aussi des probabilités. Ceci implique que la fiabilité (resp. la disponibilité), des services en séquence (PC1), est le produit des fiabilités (resp. des disponibilités) de chacun des services. Nous supposons ici que les services en séquence sont indépendants les uns des autres. Cette hypothèse est vraie du fait de la nature distribuée des architectures orientées services et en particulier grâce à la propriété de faible couplage entre les services. De la même manière, pour la mise en parallèle, la fiabilité des services en parallèle est le produit des fiabilités de chacun d'entre eux. Dans la suite, nous allons étudier les règles d'agrégation de l'attribut qualité de fiabilité sachant que les règles d'agrégation de la disponibilité sont les mêmes. Les règles d'agrégation pour les patrons de composition PC1, PC2, PC3, et PC4 sont issues de [Rosenberg, 2009]. Pour le patron de composition PC5, comme nous l'avons vu précédemment, il faut considérer le service succédant au patron de composition. Dans ce sens, nous proposons la règle d'agrégation suivante pour ce patron PC5 :

$$q_{rel}(PC5) = \prod_{i=1}^{n} q_{rel}(s_i).\Big(q_{rel}(s_B)\Big)^n \ . \tag{3.9}$$

Sachant que [Coppolino et al., 2007] ont pris en compte ce patron de composition, leur règle d'agrégation est plus complexe que la règle que nous avons proposée ; pourtant les deux règles donnent le même résultat. En effet, bien que [Coppolino et al., 2007] aient pris en compte les patrons de composition PC4, CP5, CP6, CP8, CP9 et PC10 (voir tableau 3.2), ils ont regroupé ces patrons en deux patrons appelés patrons de fiabilité qui sont *Synchronizing Parallel* et *Multi-Merge Parallel* (voir tableau 2.3). Le patron *Synchronizing Parallel* englobe les patrons PC4, PC6, PC8 et PC10 alors que le patron *Multi-Merge*

Parallel inclut les patrons PC5 et PC9. Par conséquent, les règles d'agrégation pour ces deux patrons de fiabilité sont assez génériques pour être relativement complexes par rapport à ce que l'on peut obtenir en choisissant la séparation des patrons de composition (au sens où nous les avons présentés dans la section 3.3.2), en particulier pour les patrons PC4, PC5, PC8 et PC9. D'autre part, dans nos travaux, nous prenons en compte, en plus de la fiabilité, le temps de réponse dont nous estimons que la généralisation des règles d'agrégation à la manière que cela a été réalisé dans [Coppolino et al., 2007], est difficile voire infaisable. Pour ces raisons, nous avons opté pour la séparation des patrons de composition et la définition de règles d'agrégation plus simples si c'est possible ; ce qui également est le cas pour les patrons PC8 et PC9.

En ce qui concerne le patron de composition PC6, nous considérons la règle d'agrégation proposée par [Cardoso et al., 2002, Coppolino et al., 2007] :

$$q_{rel}(PC6) = \sum_{i_1=0}^{1} \dots \sum_{i_n=0}^{1} u\left(\sum_{j=1}^{n} \delta(i_j - 1) - m\right) \prod_{j=1}^{n} \Big(\delta(i_j - 1) q_{rel}(s_j) + \delta(i_j)(1 - q_{rel}(s_j)\Big) \tag{3.10}$$

$$\text{avec } u(x) = \begin{cases} 1 & si\ x \geq 0 \\ 0 & si\ x < 0 \end{cases} \quad \text{et } \delta(x) = \begin{cases} 1 & si\ x = 0 \\ 0 & si\ x \neq 0 \end{cases}$$

La règle d'agrégation additionne tous les états probabilistes pour lesquels au moins m services s'exécutent avec succès. La somme sur $i_1, \dots, i_n$ génère tous les états probabilistes. Chaque état probabiliste est représenté par une séquence binaire $(i_1, \dots, i_n)$, où $i_j = 0$ (respectivement $i_j = 1$) représente l'échec (respectivement le succès) du service s_j. Dans chaque terme de la somme, selon la valeur de i_j, la fonction δ permet de sélectionner un seul terme du produit. Un état probabiliste est considéré seulement si le nombre de services exécutés avec succès est supérieur ou égal à m, c'est-à-dire :

$$\sum_{j=1}^{n} \delta(i_j - 1) \geq m \Leftrightarrow \sum_{j=1}^{n} \delta(i_j - 1) - m \geq 0 \ . \tag{3.11}$$

Les états qui ne satisfont pas la condition précédente sont éliminés par la fonction u.

Exemple : prenons l'exemple de trois services en parallèle dont deux sont requis pour la convergence (*AND-2/3*). L'application de la règle d'agrégation donne :

$$\begin{aligned} q_{rel}(s_1, s_2, s_3) = {} & (1 - q_{rel}(s_1)) q_{rel}(s_2) q_{rel}(s_3) + q_{rel}(s_1)(1 - q_{rel}(s_2)) q_{rel}(s_3) \\ & + q_{rel}(s_1) q_{rel}(s_2)(1 - q_{rel}(s_3)) + q_{rel}(s_1) q_{rel}(s_2) q_{rel}(s_3) \ . \end{aligned}$$

Le patron de composition PC7 est un cas particulier du patron PC6 où $m = 1$. Par conséquent, la règle d'agrégation se réduit à :

$$q_{rel}(PC7) = 1 - \prod_{i=1}^{n} (1 - q_{rel}(s_i)) \ . \tag{3.12}$$

Tableau 3.4 – Règles d'agrégation de la fiabilité et de la disponibilité

Patrons de composition	**Règles d'agrégation**
PC1- Séquence	$\prod_{i=1}^{n} q_{rel}(s_i)$
PC2- Boucle	$\Big(q_{rt}(s_i)\Big)^{l}$
PC3- XOR-XOR	$\sum_{i=1}^{n} p_i.q_{rel}(s_i)$
PC4- AND-AND	$\prod_{i=1}^{n} q_{rel}(s_i)$
PC5- AND-AND (sans synch)	$\prod_{i=1}^{n} q_{rel}(s_i).\Big(q_{rel}(s_B)\Big)^{n}$
PC6- AND-m/n	$\sum_{i_1=0}^{1}\cdots\sum_{i_n=0}^{1} u\left(\sum_{j=1}^{n}\delta(i_j-1)-m\right)$ $\prod_{j=1}^{n}\Big(\delta(i_j-1)q_{rel}(s_j)+\delta(i_j)(1-q_{rel}(s_j)\Big)$
PC7- AND-XOR	$1-\prod_{i=1}^{n}(1-q_{rel}(s_i))$
PC8- OR-OR	$\sum_{i=1}^{2^n-1} p_i \prod_{s\in\mathscr{S}_i} q_{rel}(s)$
PC9- OR-OR (sans synch)	$\sum_{i=1}^{2^n-1} p_i \prod_{s\in\mathscr{S}_i} q_{rel}(s).\Big(q_{rel}(s_B)\Big)^{\lvert\mathscr{S}_i\rvert}$
PC10- OR-m/n	$\sum_{k=1}^{N} p_k\Big(\sum_{i_1=0}^{1}\cdots\sum_{i_{\lvert\mathscr{S}_k\rvert}=0}^{1} u\left(\sum_{j=1}^{k}\delta(i_j-1)-m\right)$ $\prod_{j=1}^{\lvert\mathscr{S}_k\rvert}\Big(\delta(i_j-1)q_{rel}(s_j)+\delta(i_j)(1-q_{rel}(s_j)\Big)\Big)$
PC11- OR-XOR	$\sum_{k=1}^{2^n-1} p_k.\Big(1-\prod_{s\in\mathscr{S}_k}(1-q_{rel}(s))\Big)$

Concernant le choix multiple avec synchronisation lors de la convergence (patron PC8), nous considérons la probabilité p_i qu'une combinaison i, parmi les (2^n-1) combinaisons possibles, soit choisie lors de la divergence. Pour une combinaison $\mathscr{S}_i=(s_1,\ldots,s_m),\ 1\leq m\leq n$, les services impliqués $s_1,\ldots,s_m$ sont exécutés en parallèle. Par conséquent, la fiabilité d'une combinaison $(q_{rel}(\mathscr{S}_i))$ est le produit des fiabilités des services dans cette combinaison :

$$q_{rel}(\mathscr{S}_i)=\prod_{s\in\mathscr{S}_i} q_{rel}(s)\ .$$

Ce qui implique que la fiabilité globale est la moyenne pondérée des fiabilités des combinaisons possibles. Ainsi, nous définissons la règle d'agrégation suivante pour le patron PC8 :

$$q_{rel}(PC8)=\sum_{i=1}^{2^n-1} p_i \prod_{s\in\mathscr{S}_i} q_{rel}(s)\ . \tag{3.13}$$

La règle d'agrégation du patron PC9 est similaire à la précédente sauf qu'il faut prendre en compte le service s_B succédant au patron de composition. En effet, pour une combinaison $\mathscr{S}_i$ donnée, le service s_B est exécuté autant de fois que le nombre de services impliqués dans $\mathscr{S}_i$, c'est-à-dire le cardinal de $\mathscr{S}_i$. Par conséquent, nous obtenons la règle d'agrégation est la suivante :

$$q_{rel}(PC9)=\sum_{i=1}^{2^n-1} p_i \prod_{s\in\mathscr{S}_i} q_{rel}(s).\Big(q_{rel}(s_B)\Big)^{\lvert\mathscr{S}_i\rvert}\ . \tag{3.14}$$

Pour le patron de composition PC10 (OR-m/n), il existe $N=\sum_{m\leq k\leq n} C_n^k$ combinaisons possibles. La fiabilité d'une combinaison $\mathscr{S}$ de services revient à calculer la fiabilité

du patron $AND - m/|\mathscr{S}|$. Nous pouvons donc déduire la règle d'agrégation du patron PC10 suivante :

$$q_{rel}(PC10) = \sum_{k=1}^{N} p_k \left(\sum_{i_1=0}^{1} \dots \sum_{i_{|\mathscr{S}_k|}=0}^{1} u\left(\sum_{j=1}^{k} \delta(i_j - 1) - m \right) \prod_{j=1}^{|\mathscr{S}_k|} \Big(\delta(i_j - 1) q_{rel}(s_j) + \delta(i_j)(1 - q_{rel}(s_j)) \Big) \right) .$$

Enfin, pour le patron PC11, la fiabilité d'une combinaison $\mathscr{S}_k$, parmi les $(2^n - 1)$ combinaisons de choix possibles, peut être obtenue en appliquant la règle d'agrégation du patron PC7. Par conséquent, la règle d'agrégation du patron PC11 devient :

$$q_{rel}(PC11) = \sum_{k=1}^{2^n - 1} p_k . \left(1 - \prod_{s \in \mathscr{S}_k} (1 - q_{rel}(s)) \right) . \tag{3.15}$$

De cette manière, les règles d'agrégation pour les onze patrons de composition (voir section 3.3.2) sont définies, et ce pour chacun des attributs qualité que nous avons considérés. Notre contribution consiste principalement en la définition des règles d'agrégation relatives au temps de réponse pour les patrons PC5 à PC 11 (voir tableau 3.2), la règle d'agrégation relative à la fiabilité et à la disponibilité pour le patron PC11 ainsi que la redéfinition des règles d'agrégation relatives à la fiabilité et à la disponibilité pour les patrons de composition PC5, PC7, PC8, PC9 et PC10.

3.3.4 Agrégation utilisant les règles de patrons de *workflow*

Dans cette section, nous supposons que nous disposons des valeurs de tous les attributs qualité $q_j(s_i),\ 1 \leq j \leq p$, pour tous les services $s_i,\ 1 \leq i \leq M$, impliqués dans l'orchestration. Ceci concerne précisément la fin de l'exécution où nous avons toutes les mesures des attributs qualité. Ces mesures proviennent de la fonction d'acquisition du système de supervision (voir figure 3.1).

D'autre part, nous supposons que les modèles d'orchestration de services sont *structurés*. Un modèle d'orchestration structuré signifie que chaque patron de divergence est suivi par un patron de convergence compatible comme nous l'avons défini dans la section 3.3.2 (ex. *AND split-AND join*) [Kiepuszewski et al., 2000]. De plus, le nombre de branches sortantes dans un patron de divergence doit être égal au nombre de branches entrantes du patron de convergence correspondant [C. Jaeger, 2007]. Dans les modèles d'orchestration structurés, les patrons de composition ne peuvent être que imbriqués ; c'est-à-dire qu'il n'y a pas de chevauchement/croisement entre les patrons de composition. Les modèles d'orchestration structurés garantissent certaines propriétés comme par exemple le non blocage [Kiepuszewski et al., 2000]. À titre d'exemple, un modèle d'orchestration non structuré qui comporte un choix multiple (*OR split*) suivi d'une convergence avec synchronisation (*AND join*) entraîne un blocage lors de l'exécution. Il existe certains pro-

duits commerciaux de description de *workflow*, comme *SAP Worfklow*[3], *WebSphere MQ Workflow*[4] ou *FLOWer* [van der Aalst and Berens, 2001], qui ne supportent que les modèles de *workflow* structurés. Bien que les modèles d'orchestration structurés garantissent le non blocage à l'exécution, cela peut être au détriment de l'expressivité. La figure 3.8 présente un exemple typique utilisé pour montrer l'expressivité des modèles d'orchestration non structurés. L'exemple décrit un fragment de modèle d'orchestration non structuré, qui n'entraîne pas de blocage à l'exécution (*well-behaved*) mais qui ne peut pas être exprimé en un modèle d'orchestration structuré [Kiepuszewski et al., 2000]. Les auteurs de ces derniers travaux évoquent la possibilité de décrire les *workflow* (orchestrations) de manière non structurée, ce qui est plus simple et plus facile pour les architectes, et d'appliquer ensuite un ensemble de transformations afin de transformer le modèle non structuré en un modèle structuré équivalent au sens fonctionnel. Les techniques de transformation se résument à la duplication des activités (services) et/ou l'utilisation de variables auxiliaires pour contrôler les choix conditionnels. Les auteurs ont étudié les origines de la non structuration du modèle d'orchestration, comme par exemple le chevauchement de deux patrons de parallélisme (voir figure 3.8b), et discuté les possibilités de transformation d'un modèle non structuré en un modèle structuré. Il s'avère que les modèles d'orchestration non structurés ne contenant pas du parallélisme mais seulement des boucles arbitraires et/ou des choix conditionnels non structurés, peuvent toujours être transformés en des modèles structurés. Par contre, les modèles non structurés contenant du parallélisme (la non structuration est due au patron de parallélisme) ne peuvent être transformés que dans certains cas (voir [Kiepuszewski et al., 2000] pour les cas de transformations possibles et les transformations correspondantes). À titre d'exemple, le modèle d'orchestration (structuré) présenté en figure 3.8c représente la transformation équivalente de celui donné en figure 3.8b en utilisant la duplication des services S_7 et S_8. Bien que la transformation en un modèle structuré soit parfois possible, lcela pourrait provoquer, à notre sens, une double prise en compte des valeurs d'attributs qualité dans l'agrégation, en particulier lorsque l'on utilise la duplication des services (voir figure 3.8c). Dans ce cas, il est nécessaire d'adapter la méthode d'agrégation et corriger l'erreur due aux services dupliqués. Cela s'inscrit dans les perspectives de cette thèse (voir chapitre 6). La méthode d'agrégation que nous présentons ci-après, est seulement compatible avec les modèles d'orchestration structurés.

Ayant préalablement modélisé l'orchestration de services grâce à des langages tels que WS-BPEL, PXL, etc, la méthode d'agrégation consiste dans un premier temps à identifier les patrons de *workflow* ou plus précisément les patrons de composition utilisés. Par exemple, en utilisant le langage d'orchestration PXL (voir section 2.2.2), le modèle d'orchestration est exprimé directement à l'aide des patrons de *workflow* (voir figure 2.3) grâce à la couche de patrons de *workflow* que possède le langage (voir figure 2.2). Par contre, une orchestration exprimée en WS-BPEL nécessite la traduction des activités structurées

3. www.sap.com
4. http://www-01.ibm.com/software/integration/wmqwf/

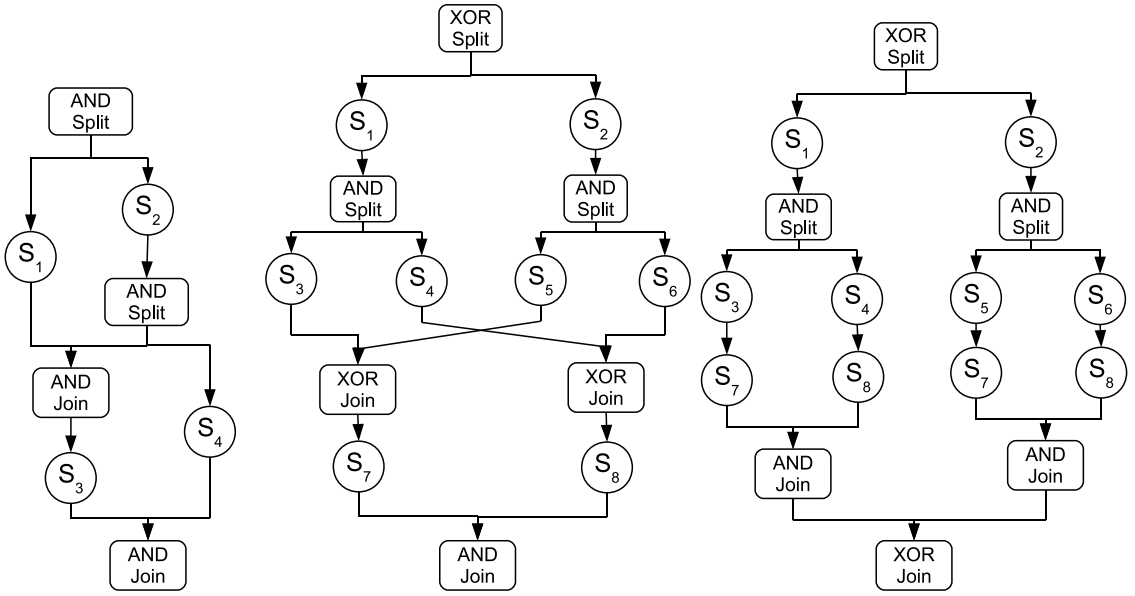


Figure 3.8 – (a) Exemple de modèle d'orchestration non structuré n'admettant pas de transformation ; (b) Non structuration due au chevauchement du parallélisme ; (c) Exemple de transformation de la non structuration

de ce langage (voir section 2.2.2) en termes de patrons de *workflow*. [Vasko and Dustdar, 2004, Hofstede et al., 2009] présentent l'équivalence des patrons de *workflow* en activités structurés de WS-BPEL. À titre d'exemple, la divergence en parallèle (*AND split*) correspond à l'activité <*flow*> en WS-BPEL. Le fait de se baser sur les patrons de *workflow* rend notre approche d'agrégation indépendante de n'importe quel langage d'orchestration (sous réserve que le modèle d'orchestration soit structuré) puisque les structures de modélisation de n'importe quel langage d'orchestration peuvent être traduites en termes de patrons de *workflow* (ou d'un sous ensemble de patrons de *workflow*).

Lorsque les patrons de composition du modèle d'orchestration sont identifiés, nous procédons à l'agrégation. Le modèle de l'orchestration est parcouru. Chaque fois qu'un patron de composition est rencontré, il est réduit en un seul noeud dont les valeurs d'attributs qualité sont déterminées par les règles d'agrégation correspondantes. La procédure est répétée jusqu'à ce que tout le modèle de l'orchestration se réduise à un seul et unique noeud. Dans chaque itération, ce sont les patrons de composition les plus imbriqués qui sont réduits. Ainsi, à la fin de la procédure d'agrégation, les valeurs des attributs qualité du noeud final (résultant de la réduction de toute l'orchestration) constituent les valeurs d'attributs qualité de toute l'orchestration $q_1(orch), \ldots, q_p(orch)$ et par suite le vecteur d'attributs qualité de l'orchestration $QoSL(orch)$. Ce vecteur représente l'une des entrées pour la deuxième phase de notre approche d'agrégation globale (voir figure 3.2).

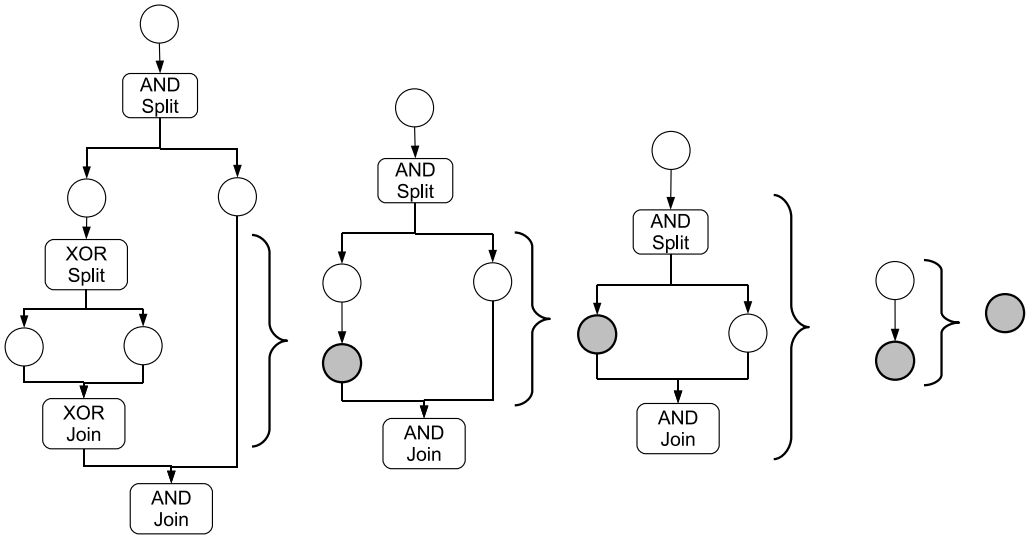


Figure 3.9 – Principe de l'agrégation

3.3.5 Synthèse

Nous avons présenté dans cette section onze patrons de composition, résultat de l'analyse des différents patrons de composition existants dans la littérature [C. Jaeger, 2007, Rosenberg, 2009, Cardoso et al., 2002, Coppolino et al., 2007, He et al., 2008]. Nous avons exprimé les onze patrons de composition recensés en se basant sur la dernière révision des patrons de *workflow* [Russell et al., 2006b] contrairement aux travaux existants qui se sont basés sur la première version des patrons de *workflow* [van der Aalst et al., 2003]. Ceci nous permet de couvrir un champs plus large de modèles d'orchestration. Ensuite, nous avons présenté les règles d'agrégation qui correspondent aux patrons de compositions pour les trois attributs qualité choisis, à savoir le temps de réponse, la fiabilité et la disponibilité. Ces règles d'agrégation concernent la fin de l'exécution où nous avons des valeurs déterministes correspondant aux mesures des attributs qualité. Notre contribution a consisté principalement en (voir tableau 3.2) :

- la définition de huit nouvelles règles d'agrégation dont sept règles sont relatives au temps de réponse et correspondent aux patrons de composition PC5 à PC11 et une règle relative à la fiabilité et à la disponibilité qui concerne le patron de composition PC11 ;
- la redéfinition de cinq règles d'agrégation moins complexes que celles proposées dans [Coppolino et al., 2007]. Ces règles sont relatives à la fiabilité et à la disponibilité et concernent les patrons de composition PC5, PC7, PC8, PC9 et PC10.

Enfin, nous avons présenté la méthode d'agrégation permettant de réduire le modèle d'orchestration à un seul noeud et en déduire par suite le vecteur d'attributs qualité de l'orchestration globale. Dans nos travaux, nous avons considéré trois attributs qualité mais la méthode d'agrégation a l'avantage de pouvoir prendre en compte d'autres attributs qualité : cela nécessite de leur définir des règles d'agrégation adéquates. En outre, la méthode d'agrégation est indépendante des langages d'orchestration du moment où les structures

de modélisation de ces langages peuvent être traduites en termes de patrons de *workflow*. Nous avons également discuté des modèles d'orchestration non structurés et la possibilité de les transformer en des modèles structurés. Toutefois, nous pensons que la méthode d'agrégation devrait être adaptée pour prendre en compte et neutraliser potentiellement l'erreur de l'agrégation due à la duplication des services. Au terme de l'analyse, notre approche d'agrégation reste générique : non restriction sur les attributs qualité à prendre en compte, généricité et indépendance par rapport au langage d'orchestration.

Les règles d'agrégation présentées dans cette section sont destinées à l'agrégation des valeurs déterministes, ce qui concerne principalement la fin de l'exécution. Dans la section suivante, nous abordons l'agrégation des attributs qualité pendant la phase de conception où nous considérons cette fois les attributs qualité comme des variables aléatoires définies par leur distribution de probabilités.

3.4 Agrégation des variables aléatoires

Le changement des valeurs des attributs qualité est une caractéristique inhérente des systèmes répartis. Par conséquent, nous estimons qu'il est plus réaliste de considérer les attributs qualité comme des variables aléatoires représentées par leurs distributions de probabilité. Dans ce cas, l'agrégation des distributions de probabilités relatives aux attributs qualité des services impliqués dans l'orchestration résulte en une seule distribution de probabilités de chaque attribut qualité portant sur l'orchestration globale, contrairement au cas déterministe où nous obtenons une seule valeur par attribut qualité. L'obtention d'une distribution de probabilités de chaque attribut qualité portant sur l'orchestration globale constitue une information plus riche et plus précise, notamment pendant la phase de conception de l'orchestration.

Dans le contexte de nos travaux, nous avons retenu trois attributs qualité qui sont le temps de réponse, la fiabilité et la disponibilité. Ces trois attributs qualité sont de nature non déterministe (voir tableau 2.1). Cependant, la fiabilité et la disponibilité présentent un cas particulier. En effet, ces attributs qualité sont représentés par des valeurs fixes (déterministes) bien qu'elles soient en elles-mêmes des probabilités. Autrement dit, le caractère non déterministe de ces attributs qualité est directement inclus dans leurs valeurs dénotant des probabilités. Par conséquent, il est inutile de représenter la fiabilité et la disponibilité par des distributions de probabilités à moins qu'elles dépendent du temps. Or, d'après la définition communément approuvée par la littérature (voir section 2.3.1), ces attributs qualité sont définis pour une période donnée. Pour ces raisons, pendant la phase de conception, nous représentons la fiabilité et la disponibilité par des valeurs déterministes (dénotant des probabilités). Quant au temps de réponse, nous le considérons comme une variable aléatoire continue définie par sa distribution de probabilités. Nous avons vu dans la section 2.4.4 que l'agrégation des distributions de probabilités continues n'est possible que dans le cas où l'on considère des distributions exponentielles négatives. Or, cela représente une restriction, notamment pour la représentation du temps de réponse

qui peut suivre d'autres distributions de probabilités continues comme la distribution t de *Student*, etc (voir section 2.4.2). Afin de prendre en compte tout type de distribution de probabilités pour les variables aléatoires continues, une solution consiste à les transformer en des variables aléatoires discrètes.

3.4.1 Manipulation des variables aléatoires continues

Dans cette section, nous allons présenter la discrétisation des variables aléatoires continues et réciproquement, la transformation des variables aléatoires discrètes en variables aléatoires continues.

Discrétisation d'une variable aléatoire continue

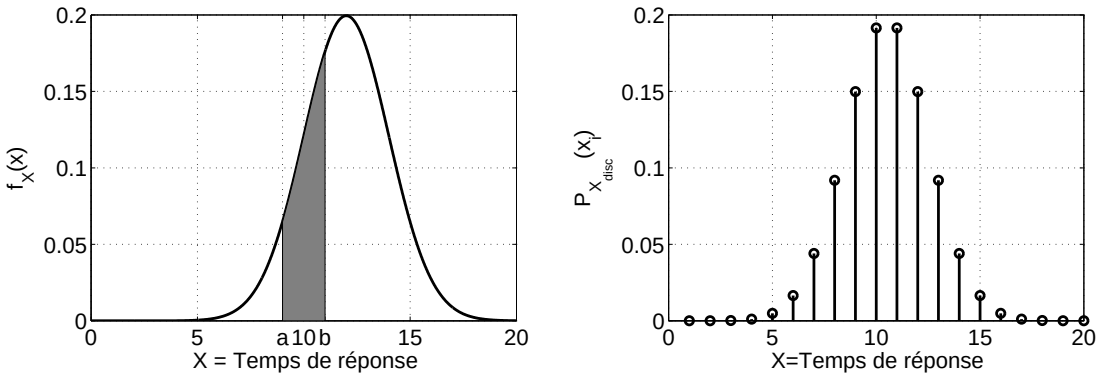
La discrétisation ou l'échantillonnage consiste ici à transformer une variable aléatoire continue, définie par sa fonction de densité de probabilité (continue) ou sa fonction de répartition (voir figure 3.10a), en une variable aléatoire discrète définie par sa loi de probabilité, c'est-à-dire des valeurs de probabilités associées à un ensemble de points (discrets) (voir figure 3.10b). Pour cela, nous subdivisons la plage de variation de la variable continue en sous intervalles. Plus le pas d'échantillonnage est petit (c'est-à-dire l'intervalle entre deux points discrets, c'est l'intervalle $[a, b]$ dans la figure 3.10a), plus la variable discrétisée est proche de la variable continue correspondante. Ensuite, nous représentons chaque sous intervalle par un nombre (ex. la borne supérieure) avec une probabilité calculée à partir de la distribution de probabilités de la variable continue. Plus exactement, l'intervalle $[a, b]$, où a et b sont des nombres réels finis, est représenté par le nombre b dont la probabilité est la valeur de l'intégrale, entre a et b, de la fonction de densité de probabilité f (partie colorée dans la figure 3.10a) :

$$P_{X_{disc}}(b) = \int_a^b f(t)dt \ . \tag{3.16}$$

Exemple : Supposons que la variable aléatoire X, représentant le temps de réponse du service s, suit une distribution de probabilités gaussienne définie par sa moyenne $\mu = 10$ et sa déviation standard $\sigma = 2$. L'expression de sa fonction de densité de probabilité est la suivante (voir figure 3.10a) :

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}} \ .$$

Prenons par exemple vingt échantillons et par suite vingt sous intervalles de la plage de variation de la variable continue X qui est approximativement entre 0 et 20 (voir figure 3.10a). Les sous intervalles sont $]0, 1],]1, 2], \ldots,]18, 19],]19, 20]$. Chaque sous intervalle est représenté par sa borne supérieure : $x_1 = 1,\ x_2 = 2,\ x_3 = 3, \ldots,\ x_{19} = 19$ et $x_{20} = 20$. Ainsi, le domaine de la variable X discrétisée, notée X_{disc}, est $Dom(X_{disc}) = \{x_1, \ldots, x_{20}\}$. La probabilité que nous associons à chaque x_i appartenant à $Dom(X_{disc})$ correspond à la



(a) (b)

Figure 3.10 – (a) Densité de probabilité gaussienne ; (b) Loi de probabilité de la variable aléatoire discrétisée

probabilité que la variable d'origine X soit dans l'intervalle que représente l'échantillon x_i, soit $]x_{i-1}, x_i]$ (avec $x_0 = 0$ dans cet exemple). En appliquant l'équation 3.16, nous pouvons calculer la loi de probabilité de X_{disc} comme suit :

$$\forall\, x_i \in Dom(X_{disc}), \quad P_{X_{disc}}(x_i) = P(x_{i-1} < X \leq x_i) = \int_{x_{i-1}}^{x_i} f(t)dt \ .$$

La figure 3.10b montre la loi de probabilité de la variable aléatoire X obtenue après discrétisation.

Nous pouvons appliquer cette méthode de discrétisation à n'importe quelle variable aléatoire continue, en particulier au temps de réponse dans le contexte de nos travaux. Voyons maintenant comment revenir à la variable aléatoire continue après qu'elle ait été discrétisée.

Transformation d'une variable discrète en une variable continue

La transformation d'une variable aléatoire discrète en une variable aléatoire continue peut être utile pour évaluer la probabilité d'un intervalle quelconque ou encore pour évaluer la méthode de discrétisation.

Pour cela, nous allons adopter la proposition de [Hwang et al., 2007]. Soit X une variable aléatoire discrète définie sur un domaine $Dom(X) = \{x_1, \ldots, x_n\}$ par sa loi de probabilité P_X. La variable X peut être vue comme une variable aléatoire continue dont la fonction de densité de probabilité f_X est obtenue de la manière suivante :

$$f_X(x) = \begin{cases} 0 & \text{si } x \leq \min(Dom(X)) \\ \frac{ceiling(x)}{ceiling(x) - floor(x)} & \text{si } \min(Dom(X)) < x \leq \max(Dom(X)) \end{cases} \tag{3.17}$$

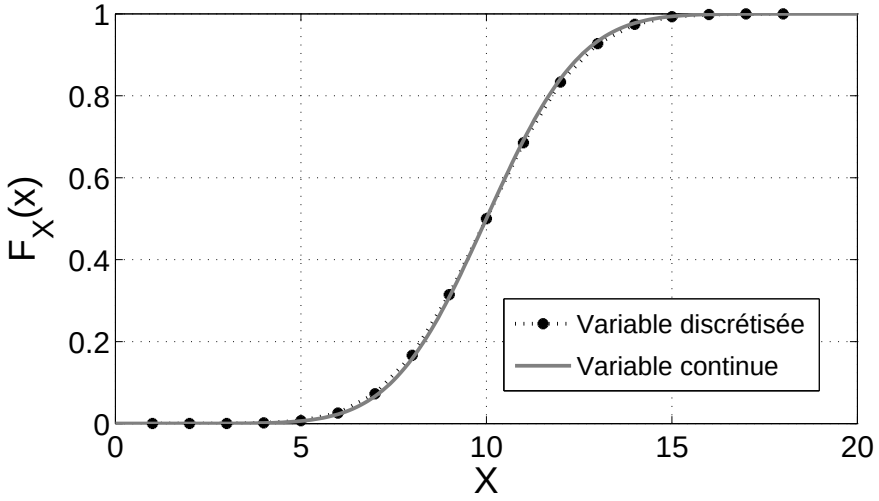


Figure 3.11 – Comparaison des fonctions de répartition d'une variable continue et d'une variable discrétisée

où :

$$ceiling(x) = \min\{y\ ;\ y \in Dom(X) \text{ et } y \geq x\}$$
$$floor(x) = \max\{y\ ;\ y \in Dom(X) \text{ et } y < x\}$$

La fonction de répartition de X est définie comme suit :

$$F_X(x) = \begin{cases} 0 & \text{si } x \leq \min(Dom(X)) \\ f_X(x)(x - floor(x)) + \sum\limits_{\substack{y<x \\ y \in Dom(X)}} P_X(y) & \text{si } \min(Dom(X)) < x < \max(Dom(X)) \\ 1 & \text{si } x \geq \max(Dom(X)) \end{cases} \tag{3.18}$$

À titre d'exemple, nous avons repris la variable aléatoire discrétisée dans l'exemple précédent dont la loi de probabilité est représentée dans la figure 3.10b et nous l'avons transformée en une variable aléatoire continue. Le résultat est donné en figure 3.11 : la courbe en trait gris continu représente la fonction de répartition de la variable aléatoire continue (d'origine) alors que la courbe en traits pointillés correspond à celle de la variable discrétisée que nous avons transformée en une variable continue par application des équations 3.17 et 3.18. Nous pouvons constater que les deux courbes sont quasiment confondues. Ceci prouve que l'on peut discrétiser les variables aléatoires continues par la méthode que nous avons présentée sans perte de généralité.

En guise de conclusion, dans la suite de cette section, nous considérons le temps de réponse comme une variable aléatoire discrète réelle et positive pour l'agrégation pendant la phase de conception. La fiabilité et la disponibilité sont bien entendu des valeurs déterministes. La méthode de réduction du modèle d'orchestration (voir figure 3.9) reste la

même que celle présentée précédemment (voir section 3.3.4). Quant aux règles d'agrégation définies dans la section 3.3.3, elles ne changent pas en ce qui concerne la fiabilité et la disponibilité. En revanche, pour le temps de réponse, les règles d'agrégation restent les mêmes sauf que les grandeurs manipulées (q_i) dans ces règles ne sont plus des valeurs déterministes dans ce cas, mais des variables aléatoires discrètes. À titre d'exemple, la variable aléatoire discrète Z représentant le temps de réponse de n services en séquence (patron PC1) dont le temps de réponse de chacun de ces services est représenté par une variable aléatoire discrète X_i est :

$$Z = \sum_{i=1}^{n} X_i \ .$$

Toutefois, il convient de présenter les opérations mathématiques utilisées dans les règles d'agrégation, pour les variables aléatoires (discrètes). Ceci inclut les opérateurs d'addition, de multiplication, de maximum, de minimum et de choix exclusif.

3.4.2 Opérations mathématiques sur les variables aléatoires discrètes

Rappelons que les services dans les AOS sont offerts par différents fournisseurs (organisations, entreprises, etc) indépendants les uns des autres. Même si les services appartiennent au même fournisseur, la propriété de faible couplage sur laquelle se base les AOS fait de sorte que l'on puisse considérer ces services comme indépendants. Ceci induit que les variables aléatoires (relatives aux attributs qualité de différents services) impliquées dans une opération mathématique sont indépendantes.

Soient X et Y deux variables aléatoires discrètes définies par leurs lois de probabilité respectives P_X et P_Y. Posons $Dom(X) = \{x_1, \ldots, x_m\}$ et $Dom(Y) = \{y_1, \ldots, y_n\}$ les domaines respectifs de X et Y. Notons par Z la variable aléatoire qui résulte de l'opération mathématique d'agrégation. Nous présentons, ci-après, la détermination de la loi de probabilité P_Z ainsi que son domaine $Dom(Z)$ pour chacune des opérations mathématiques suivantes :

Addition de deux variables aléatoires indépendantes ($Z = X + Y$)

Le domaine de Z est $Dom(Z) = \{z_1, \ldots, z_k\}$, $\max(m, n) \leq k \leq m.n$, obtenu en additionnant chaque élément de $Dom(X)$ avec chaque élément de $Dom(Y)$. La probabilité de $z_i \in Dom(Z)$ est calculée de la manière suivante :

$$P_Z(z_i) = \sum_{\substack{x+y=z_i \\ x \in Dom(X) \\ y \in Dom(Y)}} P_X(x).P_Y(y) \ .$$

Exemple : Soient X et Y deux variables aléatoires discrètes avec $Dom(X) = \{4, 8\}$ et $Dom(Y) = \{8, 12\}$. Soient $P_X(4) = 0.4$, $P_X(8) = 0.6$, $P_Y(8) = 0.3$ et $P_Y(12) = 0.7$. La

variable aléatoire Z tel que $Z = X + Y$ est définie par :

$$Dom(Z) = \{12, 16, 20\}$$

$$P_Z(12) = 0.4 \times 0.3 = 0.12 \qquad P_Z(20) = 0.6 \times 0.7 = 0.42$$

$$P_Z(16) = 0.4 \times 0.7 + 0.6 \times 0.3 = 0.46$$

Multiplication de deux variables aléatoires indépendantes ($Z = X.Y$)

Le domaine de Z est $Dom(Z) = \{z_1, \ldots, z_k\}$, $\max(m, n) \leq k \leq m.n$, obtenu en multipliant chaque élément de $Dom(X)$ avec chaque élément de $Dom(Y)$. La probabilité de $z_i \in Dom(Z)$ est calculée de la manière suivante :

$$P_Z(z_i) = \sum_{\substack{x.y=z_i \\ x \in Dom(X) \\ y \in Dom(Y)}} P_X(x).P_Y(y) \ .$$

Maximum de deux variables aléatoires indépendantes ($Z = max(X, Y)$) :

Le domaine de Z est $Dom(Z) = \{z_1, \ldots, z_k\}$, $\min(m, n) \leq k \leq m + n$, obtenu en fusionnant les éléments de $Dom(X)$ avec les éléments de $Dom(Y)$:

$$Dom(Z) = Dom(X) \cup Dom(Y) \ .$$

La probabilité de $z_i \in Dom(Z)$ est calculée de la manière suivante :

$$P_Z(z_i) = \begin{cases} P_X(z_i).\sum\limits_{\substack{y<z \\ y \in Dom(Y)}} P_Y(y) & si\ z_i \in Dom(X)\ et\ z_i \notin Dom(Y) \\ P_Y(z_i).\sum\limits_{\substack{x<z \\ x \in Dom(X)}} P_X(x) & si\ z_i \in Dom(Y)\ et\ z_i \notin Dom(X) \\ P_X(z_i).\sum\limits_{\substack{y \leq z \\ y \in Dom(Y)}} P_Y(y) + P_Y(z_i).\sum\limits_{\substack{x<z \\ x \in Dom(X)}} P_X(x) & si\ z_i \in Dom(X)\ et\ z_i \in Dom(Y) \end{cases} .$$

Minimum de deux variables aléatoires indépendantes ($Z = min(X, Y)$) :

Le domaine de Z est $Dom(Z) = \{z_1, \ldots, z_k\}$, $\min(m, n) \leq k \leq m + n$, obtenu en fusionnant les éléments de $Dom(X)$ avec les éléments de $Dom(Y)$:

$$Dom(Z) = Dom(X) \cup Dom(Y) \ .$$

La probabilité de $z_i \in Dom(Z)$ est calculée de la manière suivante :

$$P_Z(z_i) = \begin{cases} P_X(z_i).\sum\limits_{\substack{y>z \\ y\in Dom(Y)}} P_Y(y) & si\ z_i \in Dom(X)\ et\ z_i \notin Dom(Y) \\ P_Y(z_i).\sum\limits_{\substack{x>z \\ x\in Dom(X)}} P_X(x) & si\ z_i \in Dom(Y)\ et\ z_i \notin Dom(X) \\ P_X(z_i).\sum\limits_{\substack{y\geq z \\ y\in Dom(Y)}} P_Y(y) + P_Y(z_i).\sum\limits_{\substack{x>z \\ x\in Dom(X)}} P_X(x) & si\ z_i \in Dom(X)\ et\ z_i \notin Dom(Y)\ . \end{cases}$$

Choix exclusif entre $X_1, \ldots, X_k$ variables aléatoires indépendantes avec des probabilités $p_1, \ldots, p_k$ ($Z = \bigoplus\limits_{1\leq i\leq k} (X_i, p_i)$) :

Le domaine de Z est $Dom(Z) = \bigcup\limits_{1\leq i\leq k} Dom(X_i)$. La probabilité de $z_i \in Dom(Z)$ est calculée de la manière suivante :

$$P_Z(z_i) = \sum_{z_i\in Dom(X_j)} p_j.P_{X_j}(z_i)\ .$$

Notons que pour le patron de *workflow* boucle, lorsque le nombre d'itérations est inconnu (voir section 3.3.2), nous le considérons comme une variable aléatoire discrète L définie par une loi de probabilité P_L sur un domaine fini $Dom(L) = \{l_1, \ldots, l_c\}$, avec l_c le nombre maximum d'itérations. À titre d'exemple, pour un nombre d'itération $l_i \in Dom(L)$, la variable aléatoire du temps de réponse qui en résulte est $X(l) = \sum_{k=1}^{l_i} X$. À l'exécution, un seul nombre d'itérations $l \in Dom(L)$ va être considéré. Par conséquent, la variable aléatoire agrégée est un choix exclusif de $X(l)$ avec des probabilités $P_L(l)$, $l_1 \leq l \leq l_c$.

Par ailleurs, nous pouvons remarquer que la taille du domaine de la variable aléatoire Z, résultant de l'opération mathématique, peut augmenter considérablement. Par exemple, la taille du domaine de Z à l'issue de l'addition de k variables aléatoires dont le domaine de chacune d'elles est de taille n, est de l'ordre de n^k. Afin de réduire la taille du domaine de la variable aléatoire après chaque opération mathématique, nous proposons d'utiliser une méthode de réduction appelée la méthode *greedy* [Hwang et al., 2007].

3.4.3 Réduction du domaine des variables aléatoires agrégées

Considérons une variable aléatoire X avec un domaine $Dom(X) = \{x_1, \ldots, x_n\}$ ($x_i < x_{i+1},\ \forall\ 1 \leq i < n$) dont nous voulons réduire la taille de n à m. Le principe de la méthode *greedy* consiste à fusionner chaque paire d'éléments adjacents (x_i, x_{i+1}) du $Dom(X)$, ayant l'erreur de fusion la plus faible, jusqu'à atteindre la taille m du domaine souhaitée. L'erreur de fusion de (x_i, x_{i+1}), notée $erreur\_fusion(x_i, x_{i+1})$, est définie de la manière suivante [Hwang et al., 2007] :

$$erreur\_fusion(x_i, x_{i+1}) = \frac{P_X(x_i)}{P_X(x_i) + P_X(x_{i+1})}.(x_{i+1} - x_i)^2\ .$$

Quand une paire (x_i, x_{i+1}) est fusionnée, x_i est éliminée et la probabilité de x_{i+1} est recalculée en ajoutant à sa probabilité initiale, celle de x_i :

$$P_X(x_{i+1}) = P_X(x_i) + P_X(x_{i+1}) \ .$$

Le nombre total d'itérations est $(n - m)$. Cet algorithme est appliqué après chaque opération mathématique. Notons que la plupart des opérations mathématiques (addition, multiplication, maximum, minimum) sont définies pour deux opérandes. Par conséquent, lorsque le patron de composition contient j services, il faut appliquer l'opération mathématique correspondante sur les variables aléatoires deux par deux. Cela nécessite $(j - 1)$ applications de l'opération mathématique. La réduction de la variable aléatoire est effectuée après chaque application de l'opération mathématique.

Exemple : Soient s_1, s_2 et s_3 trois services en séquence. Supposons que X_1, X_2 et X_3 sont les variables aléatoires continues représentant le temps de réponse de ces services. Après la discrétisation, le domaine de chacune des variables est constitué d'un ensemble de points (valeurs), soit vingt valeurs. Rappelons que la règle d'agrégation du temps de réponse pour le patron séquence est la somme des temps de réponse des services concernés (voir section 3.3.3) : $X_1 + X_2 + X_3$. Pour calculer cette somme, il faut tout d'abord calculer la somme $Z = X_1 + X_2$. Le domaine de la variable aléatoire résultante Z dans ce cas peut atteindre la taille de quatre cents. Il faut donc réduire ce domaine à la taille de vingt. Ensuite, on calcule la somme $Z + X_3$ et on effectue de nouveau la réduction.

3.4.4 Synthèse

Nous avons présenté dans cette section les étapes requises pour l'application des règles d'agrégation dans le cas où les attributs qualité sont pris comme des variables aléatoires. Ces étapes comprennent la discrétisation des variables aléatoires continues, l'application des opérateurs mathématiques (impliqués dans les règles d'agrégation) aux variables aléatoires et la réduction du domaine des variables aléatoires agrégées. Dans notre contexte, nous avons une seule variable aléatoire continue qui est le temps de réponse. La fiabilité et la disponibilité sont par nature des valeurs fixes.

Par ailleurs, la plupart des travaux considèrent les moyennes statistiques comme estimations des valeurs d'attributs qualité (en particulier pour le temps de réponse) lors de la phase de conception [Rosenberg, 2009] ou encore au cours de l'exécution (pour les services qui ne sont pas encore exécutés) [Canfora et al., 2008, Szydlo and Zielinski, 2008]. Dans notre approche, nous considérons les distributions de probabilités des temps de réponse, relatifs aux services impliqués dans l'orchestration, pour avoir une information plus riche et plus précise. En effet, prenons l'exemple de deux services en parallèle (patron de composition PC4, voir section 3.3.2). Le temps de réponse du service s_1 (respectivement du service s_2) suit une distribution gaussienne de moyenne 60 (respectivement 80) et de déviation standard 20 (respectivement 25). Si nous considérons seulement les moyennes, le temps de

réponse globale du patron de composition est (voir section 3.3.3) $max(60, 80) = 80$. Nous avons appliqué la méthode d'agrégation présentée dans cette section en prenant comme plage de variation l'intervalle $[0, 1000]$ et un nombre d'échantillons égale à vingt. Dans ces conditions, nous avons obtenu une moyenne de la distribution de probabilités de la variable agrégée égale à 111. Nous constatons donc une erreur d'agrégation de l'ordre de 28%, ce qui est non négligeable. Cette erreur peut encore s'amplifier avec le nombre de services et les patrons de composition impliqués dans l'orchestration. Par conséquent, les décisions prises et par suite les actions de correction à mener sur l'orchestration peuvent ne pas être adéquates.

Dans la section suivante, nous présentons l'agrégation des variables aléatoires mais au cours de l'exécution de l'orchestration.

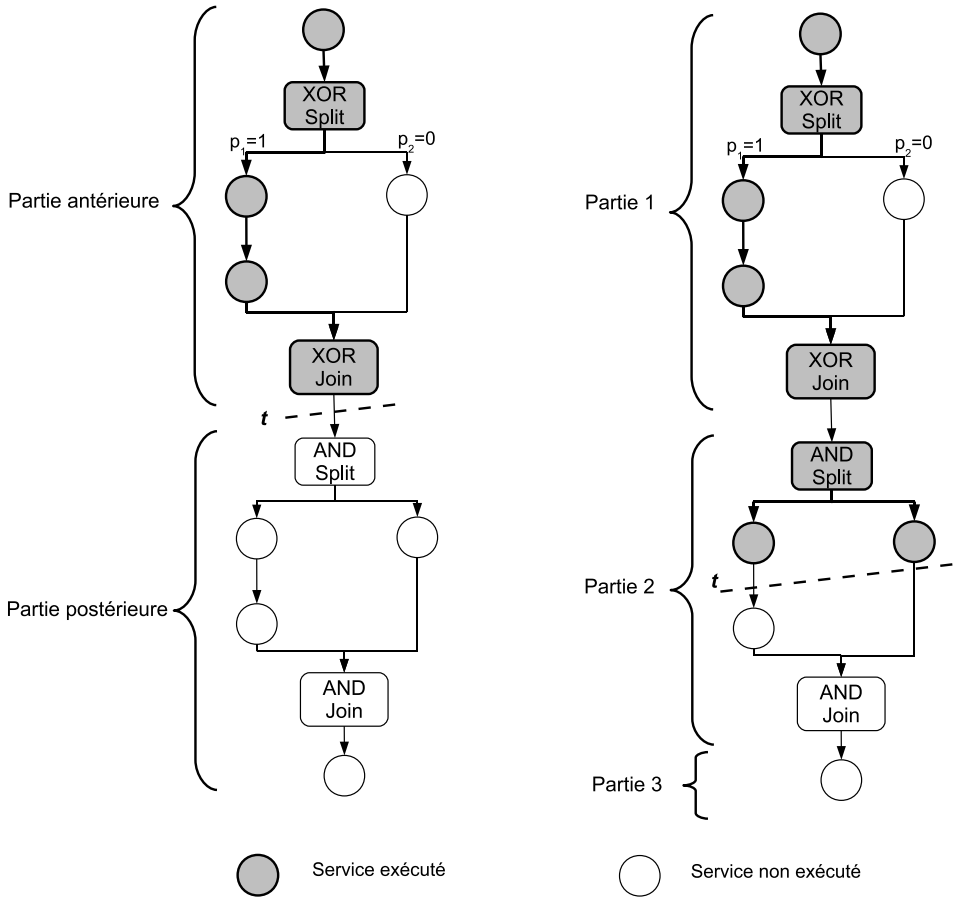
3.5 Agrégation des attributs qualité au cours de l'exécution

Nous avons vu jusqu'ici comment agréger *(i)* les valeurs déterministes correspondant notamment à l'agrégation à la fin de l'exécution (voir section 3.3) et *(ii)* les variables aléatoires correspondant à l'agrégation pendant la phase de conception (voir section 3.4). Dans cette section, nous nous intéressons à l'agrégation des attributs qualité au cours de l'exécution de l'orchestration. Dans ce cas, nous avons d'une part des mesures (valeurs déterministes) de tous les attributs qualité de la partie exécutée de l'orchestration, et d'autre part, des valeurs déterministes et/ou des variables aléatoires relatives aux attributs qualité de la partie non exécutée de l'orchestration d'autre part. Dans ce contexte, nous pouvons distinguer deux situations selon la position dans l'instance [5] de l'orchestration à l'instant t de l'évaluation. Dans la première situation, l'instant t de l'évaluation divise l'instance d'orchestration en deux parties qui peuvent être mises en séquence. Dans la deuxième situation, l'instant t de l'évaluation se trouve à l'intérieur d'un patron de composition. Voyons maintenant comment traiter ces deux situations. Notons que le traitement que nous présentons ci-après pour ces deux situations concernent particulièrement l'agrégation des variables aléatoires, le temps de réponse dans notre cas. La distinction de ces deux situations ne se présente pas pour l'agrégation des valeurs déterministes (c'est-à-dire de la fiabilité et de la disponibilité dans notre cas) au cours de l'exécution. Dans ce cas l'agrégation au cours de l'exécution s'effectue de la même manière que pendant la phase de conception et à la fin de l'exécution (voir section 3.3) en ce qui concerne la fiabilité et la disponibilité.

3.5.1 Situation 1 : l'instant t divise l'orchestration en deux parties en séquence

Dans cette situation, chacune des deux parties, séparées par l'instant t, peut contenir un ou plusieurs services et/ou un ou plusieurs patrons de composition (voir figure 3.12a). Nous

5. L'invocation d'une opération de service ou d'une orchestration de services (vu comme un service composite) donne naissance à une instance (exécution) de service (ou d'orchestration de services)



(a) L'instant t est en dehors d'un patron de composition

(b) L'instant t est à l'intérieur d'un patron de composition

Figure 3.12 – Différents instants d'évaluation des attributs qualité au cours de l'exécution

appelons ces deux parties : partie antérieure et partie postérieure à l'instant t. Ces deux parties sont traitées séparément. La partie antérieure est agrégée moyennant les règles d'agrégation déterministes. Ainsi, les valeurs considérées sont les mesures des attributs qualité. Le nombre d'itérations est le nombre réellement exécuté. Enfin, pour les patrons de composition de choix exclusif (PC3) ou de choix multiple (PC8 à PC11), la probabilité p_i du service ou de la combinaison de services choisie est mise à un et les autres sont mises à zéro. Le résultat de l'agrégation de cette partie (antérieure) est des valeurs déterministes V_i, $1 \leq i \leq p$, pour chacun des attributs qualité i considérés. La partie postérieure, quant à elle, est agrégée en appliquant la méthode pour les variables aléatoires. Ainsi, nous obtenons une seule variable aléatoire X_i par attribut qualité i. Ensuite, nous agrégeons les deux parties, c'est-à-dire pour un attribut qualité i nous agrégeons la valeur déterministe V_i avec la variable aléatoire X_i. La valeur déterministe V_i peut être vue comme une variable aléatoire Y_i dont le domaine est restreint à la seule valeur V_i ($Dom(Y_i) = \{V_i\}$) et sa loi de probabilité est $P_{Y_i}(V_i) = 1$. Les parties antérieure et supérieure étant toujours en séquence, le résultat de l'agrégation est une variable aléatoire Z_i par attribut qualité i, obtenue en appliquant la règle d'agrégation du patron séquence (PC1) sur les variables aléatoires X_i et Y_i. Ainsi, pour le temps de réponse, on a :

$$Dom(Z_i) = \{x_j + V_i,\ \forall\, x_j \in Dom(X_i)\}$$

$$P_{Z_i} = P_{X_i}\ .$$

3.5.2 Situation 2 : l'instant t est à l'intérieur d'un patron de composition

Lorsque l'instant t de l'évaluation se trouve à l'intérieur d'un patron de composition (voir figure 3.12b), nous divisons le modèle de l'orchestration en trois parties de la manière suivante : la première partie commence du début de l'exécution jusqu'à la fin du dernier patron de composition précédent l'instant t (voir figure 3.12b). La deuxième partie est le patron de composition qui est en cours d'exécution tandis que la troisième partie est constituée par le reste du modèle de l'orchestration. La première et la troisième partie sont traitées de la même manière que précédemment en les considérant respectivement comme partie antérieure et partie postérieure. La deuxième partie est traitée selon le patron de composition qui est en cours d'exécution :

Boucle : rappelons que le nombre d'itérations est représenté par une variable aléatoire discrète L dont le domaine est $Dom(L) = \{l_1, \ldots, l_c\}$ et la loi de probabilité est P_L. Supposons qu'à l'instant t de l'évaluation, il y a l_{exec} itérations qui ont été exécutées telles que :

$$\exists\ 1 \leq i \leq c,\ l_{exec} \leq l_i\ .$$

Dans ce cas, le nombre d'itérations restant peut être $l_i - l_{exec}, \ldots, l_c - l_{exec}$. Nous supposons ici que le nombre d'itérations est *sans vieillissement*, c'est-à-dire que le nombre d'itérations au delà de l'instant t est indépendant de l'instant t. Notons $L^{'}$ une nouvelle variable

aléatoire représentant le nombre d'itérations restant. Le domaine de $L^{'}$ est :

$$Dom(L^{'}) = \{l_i^{'}, \ldots, l_c^{'}\}, \text{ avec } \forall\ i \leq k \leq c\,,\ l_k^{'} = l_k - l_{exec}\ .$$

Nous définissons la loi de probabilité de $L^{'}$ comme suit :

$$\forall\ i \leq k \leq c\,,\ P_{L'}(l_k^{'}) = P_{L'}(l_k - l_{exec}) = \frac{P_L(l_k)}{\sum_{j=i}^{c} P_L(l_j)}\ .$$

Ainsi, le patron boucle global peut être vu comme deux boucles mises en séquence : la première boucle possède un nombre d'itérations l_{exec} déterministe tandis que la seconde boucle possède un nombre d'itérations aléatoire défini par la variable aléatoire $L^{'}$. Par conséquent, en agrégeant chacune des boucles par la méthode d'agrégation convenable, nous obtenons respectivement une valeur déterministe V_i et une variable aléatoire X_i par attribut qualité i. Ceci revient exactement au cas de la situation 1 (voir section 3.5.1)

Parallélisme : le parallélisme implique les patrons de composition de PC4 à PC7. Nous supposons dans un premier temps, que chaque branche de la mise en parallèle contient un seul service. Il se peut qu'à l'instant t, il y ait des services qui ont terminé leur exécution et d'autres qui sont encore en cours d'exécution. Dans ce cas, nous appliquons le même principe que précédemment. Les attributs qualité des services exécutés sont représentés par des variables aléatoires Y_i dont le domaine $Dom(Y_i) = \{V_i\}$, tel que $P_{Y_i}(V_i) = 1$, et les attributs qualité des autres services (en cours d'exécution) sont des variables aléatoires X_i (connues). Ensuite, nous appliquons la règle d'agrégation du patron correspondant pour chaque attribut qualité. Nous obtenons ainsi, une variable aléatoire Z_i par attribut qualité i pour tout le patron de composition.

Dans le cas où une branche de la mise en parallèle contient plusieurs services en séquence (voir figure 3.12b), il convient d'agréger tout d'abord les services en séquence. Ceci revient à la situation 1 où le modèle d'orchestration considéré se limite à la branche de la mise en parallèle. En général, si un patron de composition est imbriqué dans un autre patron de composition, l'agrégation commence par le patron le plus imbriqué vers le moins imbriqué (voir section 3.3.4).

Choix : le choix implique le choix exclusif (PC3) ou le choix multiple (PC8 à PC11). Le choix est effectué au début de la divergence (avant l'instant t). Par conséquent, la probabilité p_i du service i choisi (pour le choix exclusif) ou de la combinaison i de services (pour le choix multiple) est égale à un (les autres probabilités sont nulles). Ainsi, le patron de choix exclusif se ramène à un patron de séquence (situation 1) et les patrons de choix multiple deviennent équivalents aux patrons de parallélisme.

En résumé, l'agrégation des attributs qualité à un instant t au cours de l'exécution fournit une distribution de probabilités pour le temps de réponse global de l'orchestration

et des valeurs fixes représentant la fiabilité et la disponibilité de l'orchestration.

3.6 Synthèse

Nous avons présenté dans ce chapitre la méthode d'agrégation par les patrons de *workflow* correspondant à la première phase de notre approche d'agrégation (voir figure 3.2) qui, elle-même s'inscrit dans le cadre d'une approche de supervision (voir figure 3.1). Cette méthode d'agrégation ne pose pas de restrictions sur les attributs qualité à prendre en compte (voir section 3.3.3) et est générique et indépendante de tout langage d'orchestration (voir section 3.3.4). Nous avons détaillé la méthode d'agrégation pendant les différentes phases du cycle de vie de l'orchestration, à savoir pendant la phase de conception, au cours de l'exécution et à la fin de l'exécution. Nous avons proposé de représenter les attributs qualité de nature non déterministe (comme le temps de réponse) par des variables aléatoires pendant la phase de conception (voir section 3.4). Ceci permet d'avoir des estimations plus précises des attributs qualité de l'orchestration. D'autre part, nous avons présenté un nouveau traitement des attributs qualité de l'orchestration au cours de son exécution, notamment pour ceux qui sont représentés par des variables aléatoires (voir section 3.5). Ceci consiste à agréger des valeurs déterministes (mesures relatives à la partie exécutée de l'orchestration) avec des variables aléatoires (estimations relatives à la partie non exécutée de l'orchestration).

D'autre part, nous avons analysé plusieurs travaux se focalisant sur l'agrégation par les patrons de *workflow* et nous en avons ressorti onze patrons de composition [Fakhfakh et al., 2012] (voir section 3.3.2). Ces patrons de composition n'ont jamais été pris en compte, ensemble, dans aucun des travaux de la littérature. En outre, nous avons décrit ces patrons de composition à l'aide des patrons de *workflow* issus du dernier bilan du WPI [Russell et al., 2006a]. Nous pensons que ceci permet de couvrir un champs plus large de langages et, par suite, de modèles d'orchestration. Ensuite, nous avons défini de nouvelles règles d'agrégation (voir section 3.3.3) : sept règles relatives au temps de réponse correspondant aux patrons de composition PC5 à PC11 et une règle relative à la fiabilité et à la disponibilité pour le patron de composition PC11 (voir tableau 3.2). Nous avons estimé que certaines règles d'agrégation relatives à la fiabilité qui ont été proposées dans [Coppolino et al., 2007], sont très complexes notamment pour les patrons de composition PC5, PC7, PC8, PC9 et PC10. De ce fait, nous avons proposé cinq règles d'agrégation plus simples pour la fiabilité et la disponibilité correspondant à ces patrons de composition. En dernier lieu, nous avons également présenté une méthode de réduction du modèle d'orchestration (voir figure 3.9) permettant de réduire tout le modèle d'orchestration en un seul et unique noeud. Selon que l'on est pendant la phase de conception, au cours de l'exécution ou à la fin de l'exécution, la méthode de réduction permet de fournir une seule variable aléatoire représentant le temps de réponse global de l'orchestration dans les deux premiers cas et une valeur fixe (mesure) dans le dernier cas (à la fin de l'exécution). En ce qui concerne la fiabilité et la disponibilité, nous avons toujours des valeurs fixes se rapportant à l'or-

chestration globale, et ce quelle que soit la période d'évaluation. En guise de conclusion, le vecteur d'attributs qualité (constitué de valeurs fixes et/ou des variables aléatoires relatives à chacun des attributs qualité) représente le résultat de la première phase de notre approche globale d'agrégation (voir section 3.2.3 et figure 3.2). Nous allons voir dans le chapitre suivant, le traitement que nous réalisions sur ce vecteur d'attributs qualité dans la deuxième phase de notre approche globale d'agrégation.

Chapitre 4

Évaluation du degré de satisfaction des orchestrations de services

4.1 Introduction

Les systèmes de supervision actuels et plus particulièrement la surveillance et la gestion de la qualité des orchestrations de services ne sont pas orientés client : ces systèmes de supervision sont basés sur des indicateurs purement techniques (attributs qualité) et ne mesurent pas le ressenti de l'utilisateur. Nous estimons que la perception par le client de la qualité de service (*Quality of Experience–QoE*) ou encore l'évaluation des attentes des clients vis-à-vis de la qualité de service perçue sont des informations qui doivent être prises en compte dans un système de supervision. Cette supervision de « haut niveau » permet de déterminer dans quelle mesure les clients sont satisfaits de la qualité des services fournis. Ceci ne remet pas en cause la nécessité d'un système de supervision technique. Ce dernier est une brique nécessaire à la supervision de haut niveau pour remonter à la source de la dégradation.

L'approche que nous proposons dans ce chapitre cible un système de supervision orienté client (section 4.2). Elle vise à construire un modèle de préférences du client pour évaluer ensuite le degré de satisfaction de l'orchestration au regard des exigences du client (section 4.3). En particulier, nous prenons en compte les dépendances préférentielles, qui peuvent éventuellement exister, dans la construction du modèle de préférences (section 4.4). Enfin, nous proposons des stratégies de surveillance et de gestion de l'adaptation dynamique permettant d'assurer la satisfaction des exigences du client (section 4.5).

4.2 Évaluation du degré de satisfaction

4.2.1 Pourquoi évaluer le degré de satisfaction?

Dans le chapitre précédent, nous avons présenté notre approche, basée sur les règles de patrons de *workflow*, pour l'agrégation des attributs qualité. Cette approche constitue la première phase de notre approche globale d'agrégation (voir figure 3.2). Elle fournit *(i)*

les distributions de chacun des p attributs qualité se rapportant à l'orchestration globale pendant sa phase de conception et au cours de son exécution et *(ii)* les valeurs (déterministes) de chacun des attributs qualité à la fin d'exécution de tous les services impliqués dans l'orchestration. Cependant, ces informations restent difficiles à interpréter (voir section 3.2.3), notamment lorsqu'il s'agit de prendre en compte plusieurs attributs qualité simultanément ($p \gg 1$). En outre, dans les systèmes de supervision « technique », l'observation se fait sur les valeurs des attributs qualité. Par conséquent, seule la surveillance des valeurs des attributs qualité par rapport aux contraintes spécifiées (dans les contrats SLA par exemple) pour détecter ou prédire l'apparition d'aléas/violations, est possible. Ceci ne permet pas d'évaluer globalement la satisfaction ou la non satisfaction du client au regard de toutes les contraintes ou encore au regard de la qualité perçue de l'orchestration globale (voir section 3.2.3). Autrement dit, *quelle est la satisfaction du client et quelle décision prendrait-il face à la qualité perçue de l'orchestration globale ?* C'est ce que nous proposons de résoudre dans ce chapitre.

La norme ISO 9001:2008 [ISO/TC 176, 2008] précise qu'un des indicateurs de performance d'un système de gestion de la qualité est la surveillance des informations relatives à la perception du client sur *le niveau de satisfaction de ses exigences.* Les informations dégagées grâce à la surveillance et à la mesure de la satisfaction du client peuvent aider les organisations à prendre des mesures visant à maintenir ou à accroître la satisfaction des clients et à atteindre leurs objectifs (avoir la confiance des clients, attirer plus de clients, générer des bénéfices commerciaux, etc).

D'autre part, la norme ISO 10004:2010 [ISO/TC 179/SC3, 2010] relative au management de la qualité et la satisfaction du client définit *la satisfaction du client* comme étant l'écart entre *les exigences du client* et *la perception du produit* (service) par le client. De ce fait, nous estimons qu'il est important d'évaluer la satisfaction des clients au regard de la qualité perçue de l'orchestration tout en prenant en considération leurs exigences. Dans ce sens, nous proposons la définition suivante du degré de satisfaction :

Définition 4.1. Le degré de satisfaction d'une orchestration de services est l'évaluation de l'assouvissement du client au regard de la qualité globale perçue de l'orchestration, et ce, par rapport à ses préférences et à ses exigences sur les attributs qualité.

[Développement Construction, 2011] précise que la mesure de la satisfaction des clients ne doit pas être réduite à un questionnaire ou une enquête remplie par le client ; comme cela est proposé généralement pour l'évaluation de l'attribut qualité « réputation ». L'enjeu essentiel réside dans l'identification de l'importance de chaque critère (attribut qualité) et sa contribution à la satisfaction globale. Autrement dit, il faut repérer les critères qui influencent la satisfaction des clients de ceux qui n'ont pas ou peu d'impact.

Dans ce contexte, nous présentons dans la section suivante une méthode permettant d'*évaluer le degré de satisfaction* des orchestrations de services au regard des exigences du client. Pour cela, nous proposons de construire *un modèle de préférences* du client

traduisant ses satisfactions vis-à-vis des valeurs d'attributs qualité (perçues) ainsi que ses préférences par rapport à l'importance des attributs qualité.

4.2.2 Agrégation utilisant une méthode d'aide à la décision multi-critères

Dans cette section, nous introduisons la démarche générale pour évaluer le degré de satisfaction avant de présenter les méthodes utilisées pour arriver à cette fin. L'évaluation de la satisfaction du client se fait par rapport à ses exigences. Pour cela, il convient tout d'abord de définir les exigences du client en termes d'attributs qualité. [Zeithaml et al., 1993] proposent deux niveaux de qualité pour définir les exigences des clients :

- *le niveau de qualité désiré* [Parasuraman et al., 1994] : il représente la satisfaction totale du client. Ce niveau de qualité est formé par les meilleures valeurs des attributs qualité qui satisfont le client. Par la suite, nous notons ce niveau de qualité par :

$$QoSL^{good} = (q_1^{good}, \ldots, q_p^{good}) \tag{4.1}$$

- *le niveau de qualité minimum acceptable* [Parasuraman et al., 1994] : il représente la limite de la satisfaction du client, c'est-à-dire au delà de ce niveau de qualité, le client est considéré non satisfait. Ce niveau de qualité est donc constitué des valeurs limites qu'accepte le client. Notons ce niveau de qualité :

$$QoSL^{neutral} = (q_1^{neutral}, \ldots, q_p^{neutral}) \tag{4.2}$$

Ces exigences constituent une entrée pour notre approche (voir figure 3.2). Généralement, les exigences des clients sont spécifiées lors de la sélection des services [Canfora et al., 2005a, Menascé et al., 2010]. Par conséquent, nous supposons qu'elles soient disponibles et accessibles (à travers les contrats SLA par exemple). L'intervalle entre le niveau de qualité désiré et le niveau de qualité minimum acceptable est appelé la zone de tolérance pouvant satisfaire le client [Parasuraman et al., 1994]. L'objectif est d'évaluer à quel point la qualité perçue de l'orchestration satisfait les exigences. Cet objectif peut être décomposé en sous-objectifs selon plusieurs critères (attributs qualité) [Clivillé, 2004]. Les sous-objectifs correspondent dans ce cas à la satisfaction de chaque attribut qualité. Le challenge réside donc à déterminer la satisfaction de chacun de ces attributs qualité et à agréger les satisfactions obtenues afin de mesurer l'atteinte de l'objectif global. Ceci revient à *une problématique d'aide à la décision multi-critères*. En effet, il existe quatre problématiques de référence dans l'aide à la décision multi-critère [Roy, 1985, chapitre 6] :

1. *la problématique du choix*, noté P.α : elle consiste en l'aide à choisir une meilleure action (alternative, option, situation, etc) ou à élaborer une procédure de sélection. Adopter cette problématique, c'est chercher à écarter le plus grand nombre d'actions dans le but de conserver, dans le cas idéal, qu'une seule action comme étant l'optimum ;
2. *la problématique du tri*, noté P.β : elle consiste en l'aide à trier les actions par

catégories. Cela consiste à affecter chaque action à une seule catégorie, les catégories étant différentes les unes des autres. Les catégories peuvent être potentiellement définies en fonction du traitement ultérieur des actions qu'elles contiennent ;

3. *la problématique du rangement*, noté P.γ : elle consiste en l'aide à ranger les actions selon un ordre de préférence décroissante. Adopter cette problématique, c'est chercher à départager les actions et les ranger en classes successives. Contrairement à la problématique précédente P.β, les classes de P.γ ne sont pas définies en préalable ; la signification de chacune des classes dépend de sa position dans le classement ;
4. *la problématique de la description*, noté P.δ : elle consiste en « la mise en évidence-d'informations relatives aux actions potentielles conçues en vue d'aider directement le décideur à les découvrir, à les comprendre, à les jauger ... » [Roy, 1985]. L'objectif lié à cette problématique vise :
   - « soit à présenter une description systématique et formalisée des actions et de leurs conséquences qualitatives ou quantitatives ;
   - soit à proposer l'adoption d'une méthodologie fondée sur une procédure cognitive convenant à une éventuelle utilisation répétitive et/ou automatisée » [Roy, 1985].

   Dans [Figueira et al., 2005], les auteurs désignent cette problématique P.δ lorsqu'il s'agit de déterminer les performances [1] des actions, parfois complétées par d'autres informations (seuils de discrimination, niveaux de rejet, poids d'importance, etc).

La problématique la plus convenable correspondant à notre contexte est la problématique P.δ, puisque nous cherchons à faciliter l'interprétation du niveau de qualité de l'orchestration (voir section 3.2.3) par le client en fournissant une seule information de haut niveau, le degré de satisfaction. En outre, nous souhaitons établir une procédure (cognitive) traduisant l'interprétation du client des niveaux de qualité, et qui va être utilisée par le système de supervision à chaque instant t de l'évaluation de la qualité de l'orchestration en exécution.

Les méthodes d'aide à la décision permettant de résoudre cette problématique et atteindre notre objectif entrent dans la catégorie de la théorie d'utilité multiattribut (*multi-attribute utility theory*) [Figueira et al., 2005, Bouyssou et al., 2006]. Les méthodes appartenant à cette catégorie tentent d'affecter une utilité (un score) à chaque niveau de qualité (action, alternative, etc). Cette utilité est un nombre réel représentant la préférabilité (degré de satisfaction) du niveau de qualité considéré ; plus la valeur est élevée, plus préférable (satisfaisant) est le niveau de qualité. La difficulté de la construction de tels scores réside dans la représentation numérique des préférences du client (décideur) à l'aide d'une fonction d'utilité globale. Dans ce cas, la décomposition de cette fonction d'utilité multiobjectif en une combinaison (souvent une somme) de fonctions d'utilité monoobjectif rend plus facile la représentation numérique des préférences du client (on parle aussi de fonctions d'utilité multiattribut et de fonction d'utilité monoattribut [Bouyssou et al.,

1. Le terme performance est employé ici dans le sens d'une information de plus haut niveau, d'une utilité globale ou encore du degré de satisfaction.

2006] ou encore de fonctions d'utilité marginales [Figueira et al., 2005]. Dans la suite, nous adoptons les appellations *(i) fonctions d'utilité marginales* pour faire référence aux utilités (satisfactions) qu'assigne chaque attribut qualité à un niveau de qualité donné, et *(ii) fonction d'utilité globale* pour faire référence au score global (degré de satisfaction) d'un niveau de qualité. La construction de ces fonctions d'utilité détermine *le modèle de préférences* du client, permettant ainsi de représenter numériquement ses préférences.

Afin de construire les fonctions d'utilité marginales et la fonction d'utilité, nous avons choisi d'appliquer une méthode d'aide à la décision qui rentre dans la catégorie de la théorie d'utilité multiattribut (permettant la décomposition de la fonction d'utilité globale). Cette méthode se base uniquement sur des informations fournies par le client pour construire les fonctions d'utilité marginales et la fonction d'utilité globale. La méthode que nous avons choisie est *Measuring Attractiveness by a Categorical-Based Evaluation TecHnique* (MACBETH) [Bana e Costa and Vansnick, 1999, Bana e Costa et al., 2005].

Dans la section suivante, nous présentons l'application de cette méthode au domaine des architectures orientées services et nous montrons ses avantages et certaines de ses limites.

4.3 Application de la méthode MACBETH

La méthode MACBETH a été développée au milieu des années 90 par C. Bana e Costa et J.C. Vansnick [Bana e Costa and Vansnick, 1999, Bana e Costa et al., 2005]. Elle est supportée par un logiciel M-MACBETH [2], développé par les auteurs de la méthode. Cette méthode est basée sur la comparaison de paires de situations (options, alternatives, actions, etc) par l'expert ou le décideur. Dans notre contexte, l'expert est celui qui a spécifié les contraintes sur les attributs qualité et a sélectionné les services impliqués dans l'orchestration. Il peut être le client direct ou l'architecte chargé de concevoir l'orchestration. Dorénavant, nous utilisons le terme client pour faire référence à la personne que l'on veut décrire un modèle de préférences. La méthode MACBETH comporte quatre étapes principales (voir figure 4.1) : la définition du contexte, la construction des fonctions d'utilité marginales, la construction de la fonction d'utilité et l'agrégation.

4.3.1 Définition du contexte

Cette étape, dans la méthode MACBETH, consiste à définir les situations [3] qui vont être présentées au client pour qu'il établisse une comparaison entre elles selon ses préférences. Nous définissons ces situations comme des niveaux de qualité de l'orchestration de services en question. Autrement dit, les situations à comparer sont des vecteurs de valeurs d'attributs qualité de l'orchestration $QoSL(orch) = (q_1(orch), \ldots, q_p(orch))$. Lorsque nous disposons d'un historique d'exécution de l'orchestration, ce sont les niveaux de qualité correspondants qui sont jugés par le client. Par exemple, pour les trois attributs qualité que

2. `www.m-macbeth.com`
3. Les situations/alternatives sont appelées *options* dans le logiciel M-MACBETH.

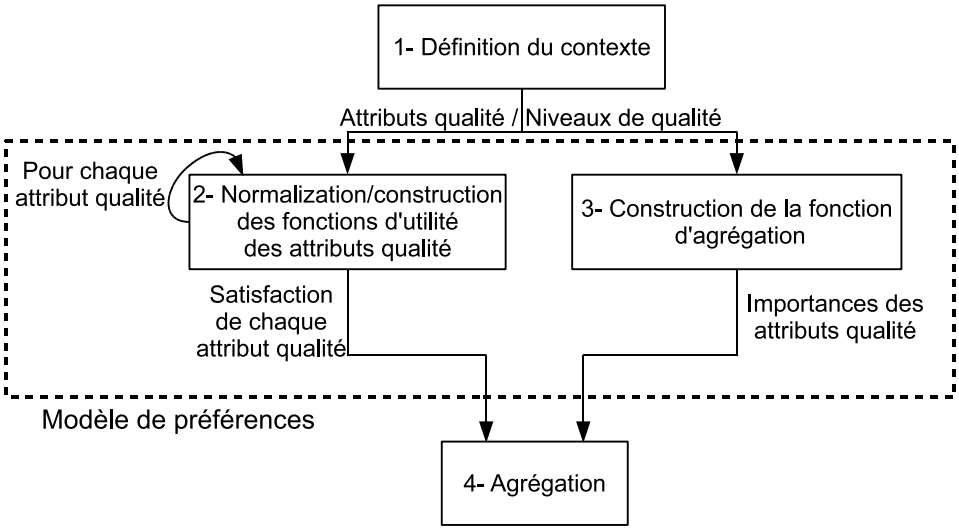


Figure 4.1 – Les principales étapes de la méthode MACBETH

nous avons sélectionnés, il nous faut trois niveaux de qualité au minimum. Dans le cas où nous ne disposons pas d'un historique d'exécution, nous pouvons constituer des niveaux de qualité fictives qui couvrent la plage de variation des attributs qualité. Notons que le niveau de qualité désiré $QoSL^{good}(orch)$ et le niveau de qualité minimum acceptable $QoSL^{neutral}(orch)$ de l'orchestration, spécifiant les exigences du client, font partie des situations à comparer. Ces deux niveaux de qualité correspondent à deux niveaux de référence dans la méthode MACBETH permettant d'assurer la *commensurabilité*. Deux utilités identiques relatives à deux attributs qualité différents sont dites commensurables, si elles donnent lieu à une même interprétation [Krantz et al., 2006]. À titre d'exemple, l'utilité 0.8 représentant l'atteinte d'une mesure du temps de réponse $q_{rt}(orch) = 120s$ à un objectif $q_{rt}^{good}(orch) = 100s$, est interprétée de la même manière que l'utilité 0.8 représentant l'atteinte d'une mesure de la fiabilité $q_{rel}(orch) = 0.9$ à un objectif $q_{rel}^{good}(orch) = 1$.

Pour assurer la commensurabilité, la méthode MACBETH spécifie, pour chaque critère (attribut qualité) i, deux niveaux de référence $q_i^{neutral}$ et q_i^{good}. Tous les $q_i^{neutral}$ (respectivement les q_i^{good}), $1 \leq i \leq p$, ont la même interprétation par le client qui est, dans notre contexte, la satisfaction minimale (respectivement la satisfaction totale) de l'attribut qualité i. Par exemple, si nous considérons l'échelle d'intervalle $[0, 1]$ pour tous les attributs qualité i, la valeur 1 correspond à la valeur $u_i(q_i^{good}(orch))$, tandis que la valeur 0 correspond à la valeur $u_i(q_i^{neutral}(orch))$.

La plupart des travaux de recherche [Menascé, 2003, Taher et al., 2005b] ont adopté l'intervalle borné [0,1] comme l'ensemble de valeurs que peut avoir la fonction d'utilité globale. Dans notre approche, nous considérons l'ensemble des nombres réels $\mathbb{R}$ comme intervalle de la fonction d'utilité globale. Cependant, nous conservons le sens d'interprétation des valeurs 0 et 1, c'est-à-dire que 0 est la satisfaction minimale et 1 et la satisfaction totale. Par conséquent, l'intervalle $]-\infty, 0[$ représente la non satisfaction du client. C'est le cas des valeurs d'attributs qualité qui sont plus mauvaises que les valeurs minimum accep-

tables $q_i^{neutral}$. Par exemple, le client fixe la valeur 0.7 comme valeur minimale acceptable pour la fiabilité et il obtient une valeur égale à 0.6 qui se traduit donc par une utilité (satisfaction) négative. L'intervalle $[0, 1]$ traduit la satisfaction du client, c'est-à-dire que les valeurs des attributs qualité sont compris dans la zone de tolérance, entre les valeurs minimum acceptables et les valeurs désirées. Enfin, l'intervalle $]1, +\infty[$ exprime la sur-satisfaction du client, c'est-à-dire que les valeurs des attributs qualité sont meilleures que les valeurs désirées par le client. À titre d'exemple, le client exprime qu'il est totalement satisfait par un temps de réponse globale égale à 500s. Supposons qu'à la fin de l'exécution de l'orchestration, le temps de réponse mesuré est égale à 450s, ceci sera traduit par une valeur normalisée (utilité) supérieure à 1.

Ayant défini le contexte, nous procédons dans la section suivante à la construction des fonctions d'utilité marginales.

4.3.2 Construction des fonctions d'utilité marginales (normalisation)

L'objectif de cette étape est de construire les fonctions d'utilité marginales, notées u_i, relatives à chaque attribut qualité i. Elle représente la première partie dans la construction du modèle de préférences. Notons que cette étape est souvent connue comme la normalisation des valeurs des attributs qualité dans la littérature des AOS [Taher et al., 2005b, Canfora et al., 2005a, hung Vu et al., 2005]. La construction des fonctions d'utilité marginales consiste à construire une échelle d'intervalle traduisant la satisfaction (du point de vue de l'utilisateur) de chaque valeur de l'attribut qualité i au regard des exigences correspondantes, c'est-à-dire q_i^{good} et $q_i^{neutral}$. Nous avons vu dans la section 2.3.1 que les attributs qualité peuvent avoir des unités de mesures différentes. De plus, ils peuvent également posséder des dimensions différentes (ascendantes ou descendantes, voir tableau 2.1). Par exemple, pour les attributs qualité choisis pour la démonstration, le temps de réponse a une dimension descendante tandis que la fiabilité et la disponibilité ont une dimension ascendante. Par conséquent, il convient de mettre à une seule échelle tous les attributs qualité pour qu'ils puissent être agrégés par la suite. Les utilités (valeurs normalisées) sont donc des grandeurs sans unité et qui possèdent toutes la même dimension (par exemple ascendante).

La construction de la fonction d'utilité marginale u_i relative à l'attribut qualité i est effectuée en interrogeant le client sur ses préférences concernant les niveaux de qualité donnés. Supposons que nous avons p niveaux de qualité $QoSL_1(orch), \ldots, QoSL_p(orch)$. La première question posée au client pour chaque paire de niveaux de qualité est :

Q1 : est-ce que l'un des deux niveaux de qualité est plus attractif que l'autre ?

Si la réponse à cette question est positive, une deuxième question est posée :

Q2 : lequel des deux niveaux est plus attractif ?

La réponse à ces deux questions pour toutes les paires de niveaux de qualité permet de les ranger par ordre de préférence selon les valeurs de l'attribut qualité i. Par exemple, si le client préfère le niveau de qualité $QoSL_j(orch)$ sur le niveau de qualité $QoSL_k(orch)$ du point de vue de l'attribut qualité i, ceci est noté :

$$QoSL_j(orch) \succ QoSL_k(orch) \ . \tag{4.3}$$

Cela signifie que le client est plus satisfait par la valeur de l'attribut qualité $q_i^j(orch)$ que par la valeur $q_i^k(orch)$, c'est-à-dire que $u_i(q_i^j(orch)) > u_i(q_i^k(orch))$.

Après avoir rangé tous les niveaux de qualité, la dernière question posée au client pour chaque paire de niveau de qualité est :

Q3 : comment jugez-vous la différence d'attractivité entre les niveaux de qualité j et k ?

Pour répondre à cette question, la méthode MACBETH propose une échelle constituée de sept niveaux pour spécifier les intensités de préférences : 0=nulle, 1= très faible, 2=faible, 3=modérée, 4=fort, 5=très fort, 6=extrême. En combinant les deux informations (préférences et intensités de préférences), nous obtenons un ensemble de relations préférentielles de la forme :

$$QoSL_j(orch) \succ^h QoSL_k(orch) \tag{4.4}$$

où h est l'intensité de préférence exprimée par le client, avec $h \in \{0, ..., 6\}$.

Exemple : prenons deux niveaux de qualité formés par les trois attributs qualité temps de réponse, fiabilité et disponibilité suivants :

$$QoSL_1(orch) = (120, 0.95, 0.9) \qquad QoSL_2(orch) = (100, 0.85, 0.98) \ .$$

Le client exprime qu'il préfère modérément ($h = 3$) le temps de réponse du niveau de qualité $QoSL_2(orch)$ à celui du niveau de qualité $QoSL_1(orch)$, c'est-à-dire qu'il juge que la différence entre 100 et 120 est modérée. Ceci est traduit par la relation de préférence suivante en ce qui concerne l'attribut qualité du temps de réponse :

$$QoSL_2(orch) \succ^3 QoSL_1(orch) \ .$$

Ces informations préférentielles (préférences et intensités de préférences [4]) sont traduites dans une matrice (voir tableau 4.1). La première colonne de la matrice est constituée par les niveaux de qualité rangés selon l'ordre de préférence du client, représenté par la fonction de permutation σ tel que $QoSL_{\sigma(1)}(orch) \succ \cdots \succ QoSL_{\sigma(p)}(orch)$. La première ligne est symétrique à la première colonne. Les cases de la matrice sont remplies par les intensités de préférences relatives à la différence d'attractivité entre le niveau de qualité de

4. Les préférences et les intensités de préférences sont appelées aussi des informations cardianles [Bana e Costa et al., 2005].

Tableau 4.1 – Préférences et intensités de préférences pour le temps de réponse

attribut qualité i	$QoSL_{\sigma(1)}(orch)$	$QoSL_{\sigma(2)}(orch)$	...	$QoSL_{\sigma(p)}(orch)$
$QoSL_{\sigma(1)}(orch)$	nulle	h_1	...	P
$QoSL_{\sigma(2)}(orch)$		nulle	...	P
$\vdots$			nulle	h_{p-1}
$QoSL_{\sigma(p)}(orch)$				nulle

$h_i \in \{0 = nulle,\ 1 = très\ faible,\ 2 = faible,\ 3 = modérée,\ 4 = fort,\ 5 = très\ fort,\ 6 = extrême\}$

la ligne correspondante et celui de la colonne correspondante. Par exemple, dans la matrice donnée par le tableau 4.1, h_1 représente l'intensité de préférence de la différence d'attractivité entre le niveau de qualité $QoSL_{\sigma(1)}(orch)$ et le niveau de qualité $QoSL_{\sigma(2)}(orch)$, c'est-à-dire $QoSL_{\sigma(1)}(orch) \succ^{h_1} QoSL_{\sigma(2)}(orch)$. Notons que la partie triangulaire inférieure de la matrice n'est pas remplie car la relation de préférence $\succ$ est asymétrique. Lorsque le client n'est pas capable d'exprimer ses intensités de préférences mais seulement ses préférences (c'est-à-dire qu'il ne peut pas répondre à la question $Q3$), ceci est noté *Positive* ou *P*. Enfin, si le client ne peut pas exprimer ses préférences (c'est-à-dire qu'il ne peut pas répondre à la question $Q1$, ceci est noté « ? ». D'autre part, l'expression des intensités de préférences sur la différence d'attractivité entre tous les couples de niveaux de qualité possibles peut s'avérer une tâche lourde et relativement difficile pour le client. Pour ces raisons et afin d'éviter les problèmes liés à des incohérences dans l'expression des intensités de préférences (inconsistance [5] des intensités de préférences) [Bana e Costa et al., 2005], nous nous limitons à l'expression des intensités de préférences entre les niveaux de qualité consécutifs après leur rangement par ordre de préférence. Autrement dit, le client remplit seulement la « seconde » diagonale de la matrice ($h_1, \ldots, h_{p-1}$ dans le tableau 4.1).

Chaque intensité de préférence exprimée par le client peut être traduite par une équation. En effet, pour un attribut qualité i :

$$QoSL_j(orch) \succ^h QoSL_k(orch) \Longleftrightarrow u_i(q_i^j(orch)) - u_i(q_i^k(orch)) = h.\alpha \qquad (4.5)$$

où α représente une graduation de l'échelle que nous cherchons à construire. Une fois que le client a exprimé toutes ses intensités de préférences, nous pouvons extraire un système d'équations de la forme de l'équation 4.5. Notons que parmi les p niveaux de qualité impliqués dans la comparaison, il y a les deux niveaux de référence $QoSL^{good}(orch)$ et $QoSL^{neutral}(orch)$. Nous avons vu précédemment (voir section 4.3.1) que les valeurs normalisées associées aux q_i^{good} et $q_i^{neutral}$ sont respectivement 1 et 0 ($u_i(q_i^{good}) = 1$ et $u_i(q_i^{neutral}) = 0$). Par conséquent, il reste $p - 2$ inconnues. Il faut donc au moins $p - 2$

5. Le logiciel M-MACBETH permet de détecter les jugements (intensités de préférences) inconsistants et propose également des suggestions pour les corriger.

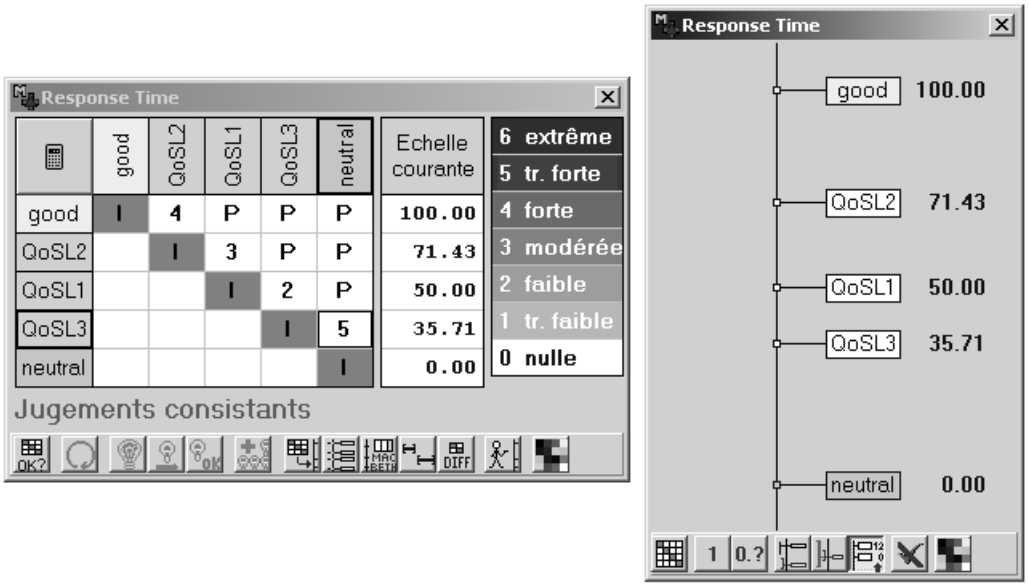


Figure 4.2 – Exemple d'une matrice de préférences et de la fonction d'utilité marginale correspondante réalisées avec le logiciel M-MACBETH

équations et par suite $p-2$ intensités de préférences exprimées par le client pour déterminer les valeurs normalisées de l'attribut qualité i de tous les niveaux de qualité restants. Ainsi, en résolvant le système d'équations, nous obtenons les valeurs normalisées $u_i(q_i^j(orch)), 1 \leq j \leq p$, de l'attribut qualité i de tous les niveaux de qualité $QoSL_k(orch)$. De cette manière, la fonction d'utilité marginale u_i relative à l'attribut qualité i est construite.

Cette démarche permet de normaliser les valeurs d'attributs qualité des p niveaux de qualité impliqués dans la comparaison. Notons que pour de nouvelles valeurs de l'attribut qualité i, nous effectuons la normalisation par interpolation linéaire. En effet, supposons que $q_i^1 < q_i < q_i^2$ et que les valeurs q_i^1 et q_i^2 ont été normalisées par la méthode MACBETH, c'est-à-dire que nous disposons des valeurs $u_i(q_i^1)$ et $u_i(q_i^2)$. La valeur $u_i(q_i)$ est déterminée par interpolation linéaire comme suit :

$$u_i(q_i) = u_i(q_i^1) + \frac{u_i(q_i^1) - u_i(q_i^2)}{q_i^1 - q_i^2} \cdot (q_i - q_i^1) \tag{4.6}$$

C'est pour cette raison que plus nous avons des niveaux de qualité à comparer, meilleure est la précision de la normalisation des nouvelles valeurs. La figure 4.2 montre un exemple de matrice de préférences du temps de réponse et sa fonction d'utilité marginale obtenue à l'aide du logiciel M-MACBETH.

La démarche présentée dans cette section est réalisée pour tous les attributs qualité i, $1 \leq i \leq p$. Nous avons supposé ici que les intensités de préférences exprimées par le client sont nettes ; c'est-à-dire que pour une paire de niveaux de qualité, le client évalue la différence d'attractivité avec une seule intensité de préférence précise. La méthode MACBETH permet également l'expression des intensités de préférences *nuancées*. Dans ce cas,

Tableau 4.2 – Préférences et intensités de préférence pour la détermination des poids d'importance des attributs qualité

	$(1,\ldots,1)$	$(1_{\sigma(1)},0_{-\sigma(1)})$	$\ldots$	$(1_{\sigma(p)},0_{-\sigma(p)})$	$(0,\ldots,0)$
$(1,\ldots,1)$	nulle	h_1	$\ldots$	P	P
$(1_{\sigma(1)},0_{-\sigma(1)})$		nulle	$\ldots$	P	P
$\vdots$			nulle	h_{p-1}	P
$(1_{\sigma(p)},0_{-\sigma(p)})$				nulle	h_p
$(0,\ldots,0)$					nulle

$h_i \in \{0 = nulle,\ 1 = très\ faible,\ 2 = faible,\ 3 = modérée,\ 4 = fort,\ 5 = très\ fort,\ 6 = extrême\}$

le client fournit un encadrement de ses intensités de préférences comme par exemple :

$$h_1.\alpha < u_i(q_i^j(orch)) - u_i(q_i^k(orch)) < h_2.\alpha$$

avec $0 < h_1 < h_2 < 6$. Ceci résulte en un système d'inéquations dont la résolution permet d'obtenir une échelle pré-cardinale [Clivillé, 2004], c'est-à-dire un encadrement de chaque valeur normalisée. Afin de passer d'une échelle pré-cardinale vers une échelle d'intervalle, il faut préciser les valeurs d'attributs qualité normalisées. Pour ce faire, le client est amené à ajuster ces valeurs à l'intérieur de l'encadrement proposé de manière à ne pas contredire ses intensités de préférence. Cette étape nécessite donc davantage l'implication du client. Afin de réduire l'implication du client, nous supposons dorénavant que les intensités de préférence exprimées par le client sont nettes (précises).

Dans cette section, nous avons présenté la démarche de construction des fonctions d'utilité marginales relatives à chacun des attributs qualité (première partie du modèle de préférences). Dans la section suivante, nous présentons la démarche pour construire la fonction d'utilité globale, deuxième partie du modèle de préférences (voir figure 4.1).

4.3.3 Construction de la fonction d'utilité globale

La fonction d'utilité globale dans la méthode MACBETH est basée sur l'opérateur de la moyenne arithmétique pondérée donnée par l'équation suivante :

$$q_{ag}(orch) = \sum_{i=1}^{p} w_i.u_i(q_i(orch))\,,\ avec\ \sum_{i=1}^{p} w_i = 1. \tag{4.7}$$

où w_i représente le poids d'importance accordé à l'attribut qualité i et reflète les préférences du client vis-à-vis des attributs qualité. Pour cette raison, la détermination des poids d'importance w_i (construction de la fonction d'utilité globale) à partir des préférences du client, fait partie du modèle de préférences. La valeur normalisée $u_i(q_i(orch))$ de l'attribut qualité i représente la satisfaction ressentie de la valeur $q_i(orch)$. Le terme

$w_i.u_i(q_i(orch))$ représente donc la contribution de la satisfaction de l'attribut qualité i à la satisfaction globale du client. Pour déterminer les poids d'importance w_i, le client est interrogé sur ses intensités de préférence sur la différence d'attractivité entre des paires de niveaux de qualité. Lors de la normalisation (voir section 4.3.2), le client compare les niveaux de qualité par rapport à la valeur de l'attribut qualité i en question et ne prend pas en compte les valeurs des autres attributs qualité. Ici, le client compare les niveaux de qualité globalement par rapport aux valeurs des p attributs qualité. Par conséquent, le poids d'importance va intervenir dans l'expression des préférences et des intensités de préférence du client. Les auteurs de la méthode MACBETH proposent de considérer des niveaux de qualité *binaires* (parfois fictifs) du type $(0_1, \ldots, 1_i, \ldots, 0_p)$ où seulement un attribut qualité i est mis à 1, les autres sont nuls. Nous notons ces situations $(1_i, 0_{-i})$ où 1_i représente la valeur désirée q_i^{good} de l'attribut qualité i, tandis que le 0_{-i}, représente la valeur minimale acceptable des attributs qualité j tel que $j \neq i$. Ces niveaux de qualité (fictifs) permettent au client de se focaliser davantage sur l'importance des attributs qualité que sur les valeurs affectées à ces attributs qualité. Toutefois, ceci n'empêche pas de considérer des niveaux de qualité avec d'autres valeurs arbitraires (plus réalistes). Nous considérons par la suite les niveaux de qualité binaires pour la détermination des poids d'importance des attributs qualité. Le degré de satisfaction (valeur agrégée) des niveaux de qualité binaires $(1_i, 0_{-i})$ est réduit à $q_{ag}(orch) = w_i$ (voir équation 4.7) puisque nous avons $u_i(q_i(orch)) = 1$ et $u_j(q_j(orch)) = 0$ pour tout $j \neq i$. De la même manière que précédemment, le client range, tout d'abord, les niveaux de qualité binaires selon son ordre de préférence. Ceci revient à ranger les poids d'importance par ordre de préférence. Ensuite, le client exprime ses intensités de préférence sur la différence d'attractivité $q_{ag}^j(orch) - q_{ag}^k(orch)$, $j \neq k$, entre chaque couple de niveaux de qualité $(1_j, 0_{-j})$ et $(1_k, 0_{-k})$ tel que $(1_j, 0_{-j}) \succ (1_k, 0_{-k})$. Les deux niveaux de référence $QoSL^{good}(orch)$ et $QoSL^{neutral}(orch)$ représentés respectivement par les vecteurs normalisés $(1, \ldots, 1)$ et $(0, \ldots, 0)$ peuvent également être impliqués dans la comparaison. L'acquisition de ces informations préférentielles se traduit par la construction de la matrice donnée en tableau 4.2. Ainsi, chaque intensité de préférence exprimée est traduite par une équation sous la forme :

$$q_{ag}^j(orch) - q_{ag}^k(orch) = h.\alpha = w_j - w_k \tag{4.8}$$

La construction de la matrice de préférence (voir tableau 4.2), telle que nous l'avons définie dans la section précédente, nous fournit $p + 1$ équations. De plus, les poids d'importance w_i doivent vérifier la condition $\sum_{i=1}^{p} w_i = 1$. Ainsi, nous obtenons un système d'équations linéaires qui peut être mis sous la forme matricielle $A \times X = B$, où :

- A est une matrice à coefficients réels ;
- X est le vecteur de paramètres inconnus de la forme $X^t = [w_1, \ldots, w_p, \alpha]$ (X^t est le transposé de X) ;
- B est un vecteur à coefficients réels de la forme $B^t = [0, \ldots, 0, 1]$.

La résolution de ce système d'équations dépend du rang de la matrice A et de ses dimen-

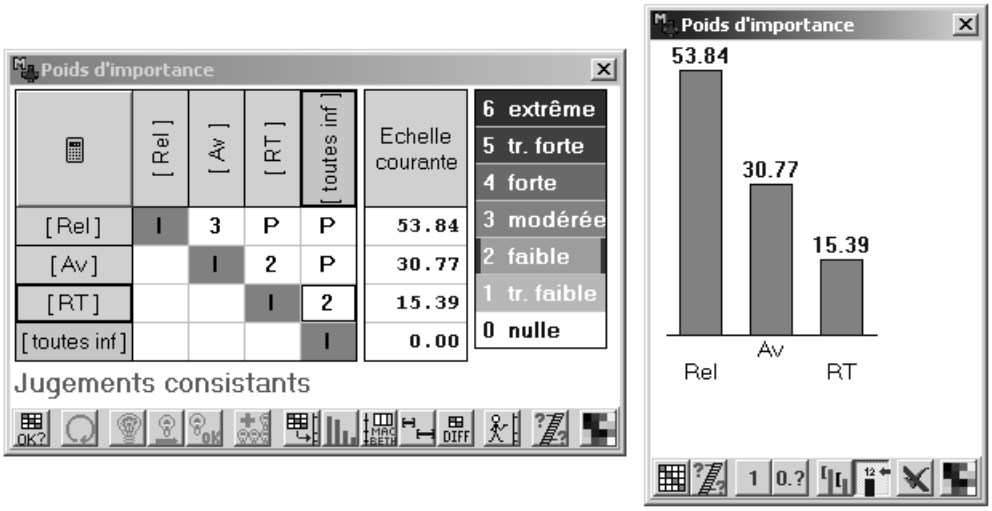


Figure 4.3 – Exemple d'une matrice de préférence pour la détermination des poids d'importance

sions (nombre d'équations extraites de la matrice des préférences et nombre de paramètres inconnus). La résolution [6] de ce système d'équations permet de déterminer les poids d'importance w_i, $1 \leq i \leq p$. La fonction d'utilité globale est ainsi construite.

La figure 4.3 montre un exemple d'une matrice de préférences pour la détermination des poids d'importance de trois attributs qualité.

4.3.4 Agrégation

Cette dernière étape consiste à agréger, par la moyenne pondérée (voir équation 4.7), la satisfaction de chaque attribut qualité i (la valeur normalisée $u_i(q_i(orch))$). Chacun des niveaux de qualité $QoSL_1(orch), \ldots, QoSL_p(orch)$, considérés dans l'étape de normalisation, va se voir attribué une valeur agrégée, un degré de satisfaction. Le client peut vérifier si les degrés de satisfaction, ainsi obtenus, traduisent bien sa satisfaction réelle ou non. Le cas échéant, la construction des fonctions d'utilité marginales et de la fonction d'utilité globale (la détermination des poids d'importance) sont reconsidérées.

Synthèse

Nous avons présenté dans cette section la construction du modèle de préférences du client en utilisant la méthode MACBETH qui se base uniquement sur l'expression des préférences et des intensités de préférences. Lorsque le modèle de préférences est valide (en conformité avec les préférences du client), tout niveau de qualité relatif à l'orchestration en question va se voir attribué un degré de satisfaction. Or, nous avons remarqué par ailleurs qu'il existe des relations entre certains attributs qualité et qu'il est possible d'avoir des

6. La résolution du système d'équations linéaires $A \times X = B$ peut être effectuée par exemple à l'aide du logiciel MATLAB en utilisant la fonction prédéfinie *mldivide* ($X = mldivide(A, B)$) qui permet, soit de fournir une solution unique dans le cas où le système est carré et est inversible, soit de donner une solution optimale au sens des moindres carrées dans le cas contraire.

dépendances entre eux. Ceci implique que les préférences du client peuvent être impactées par ces relations/dépendances entre les attributs qualité. Par conséquent, la satisfaction du client elle-même peut également être influencée par les dépendances entre les attributs qualité. Ainsi, nous estimons que si nous ne prenons pas en compte ces dépendances, le degré de satisfaction obtenu par notre approche risque de ne pas être conforme à la satisfaction réelle du client. La méthode MACBETH ne permet pas de prendre en compte ces dépendances entre les attributs qualité. En effet, la fonction d'utilité globale de cette méthode est basée sur l'opérateur d'agrégation de la moyenne arithmétique pondérée qui ne permet pas de prendre en compte des dépendances entre les attributs qualité, susceptibles d'avoir lieu [Fakhfakh et al., 2011b,Fakhfakh et al., 2011d]. Dans la section suivante, nous présentons les formes de dépendances entre les attributs qualité et nous montrons comment elles peuvent être prises en compte dans la mesure du degré de satisfaction.

4.4 Prise en compte des dépendances entre les attributs qualité

4.4.1 Dépendances entre les attributs qualité

Les critères de décision peuvent présenter certaines dépendances appelées aussi interactions [Marichal, 99, Grabisch and Labreuche, 2005]. Ce phénomène peut être très complexe et difficile à identifier dans l'aide à la décision. C'est pourquoi il est souvent négligé dans la pratique [Marichal, 99]. Nous avons recensé deux formes de dépendances entre les critères [Marichal, 99, Roy, 2009] que nous pouvons projeter dans le contexte de nos travaux :

Dépendances structurelles [Roy, 2009] : cette forme de dépendance est la plus intuitive et la mieux cernée par les décideurs. Elle est due à la présence de facteurs susceptibles d'influencer simultanément plusieurs critères. Cette dépendance peut être mise en évidence en examinant les liens entre les critères ou en identifiant les données communes entre eux [Roy, 2009]. À titre d'exemple, nous pouvons observer une corrélation entre les attributs qualité « temps de réponse » et « latence » puisque la latence fait partie du temps de réponse (voir section 2.3.1). Ainsi, ces deux attributs qualité présentent une certaine redondance. Dans notre approche, nous cherchons à construire un modèle de préférence du client afin de mesurer au mieux sa satisfaction. Par conséquent, nous considérons que si cette forme de dépendance existe, elle sera répercutée sur les mesures des attributs qualité. Autrement dit, les liens éventuels entre les attributs qualité sont traduits dans les mesures et sont dans ce cas en quelque sorte pris en compte implicitement. D'autre part, nous n'agissons pas sur les attributs qualité des services car ces derniers ne sont pas maîtrisables par le client (consommateur de service). Cette forme de dépendance est intéressante à prendre en compte par les fournisseurs de services lorsqu'ils souhaitent améliorer tel ou tel attribut qualité. Par contre, du point de vue client (consommateur de

services), l'identification de cette forme de dépendance n'a pas d'intérêt, d'où la non prise en compte des dépendances structurelles dans notre approche.

Dépendances préférentielles [Roy, 2009, Marichal, 99] : cette forme de dépendance est généralement plus difficile à comprendre par le décideur. Elle est présente dans certains modes de raisonnements qui font intervenir plusieurs critères et qui doivent être pris en compte dans la modélisation des préférences du décideur. Supposons que deux critères i et j sont dépendants au sens des préférences. Nous pouvons distinguer plusieurs cas parmi lesquels nous citons :

- l'auto-renforcement [Roy, 2009] : appelé aussi synergie positive [Marichal, 99]. Dans cette situation, la contribution conjointe des critères i et j à la satisfaction globale est plus significative que la somme des contributions de chacun de ces critères ;
- l'auto-affaiblissement [Roy, 2009] : appelé aussi synergie négative [Marichal, 99]. Dans cette situation, la contribution conjointe des critères i et j à la satisfaction globale est moins significative que la somme des contributions de chacun de ces critères ;
- l'interchangeabilité et la complémentarité [Marichal, 99] : l'interchangeabilité entre deux critères i et j se manifeste lorsque la satisfaction d'un seul critère produit presque le même effet que la satisfaction des deux. L'importance de la paire $\{i, j\}$ est donc très proche de celle de chacun des critères i et j. Dans ce cas, les critères i et j sont substitutifs ou interchangeables. Par exemple, nous pouvons constater que les attributs qualité *fiabilité* et *précision* sont interchangeables (voir section 2.3.1). La complémentarité entre deux critères i et j, quant à elle, est mise en évidence lorsque la satisfaction d'un seul critère produit très peu d'effet par rapport à la satisfaction conjointe des deux critères. Par exemple, les attributs qualité *latence* et *temps d'exécution* peuvent être vus comme des attributs qualité complémentaires.

Il existe un autre cas de dépendance préférentielle qui fait intervenir au moins trois critères : deux critères i et j sont dits dépendants au sens des préférences d'un troisième critère k si la façon de comparer deux situations a et b est susceptible d'être influencée par la satisfaction du critère k dans les situations a et b [Roy, 2009]. Autrement dit, bien que la satisfaction du critère k soit identique pour les situations a et b, selon cette satisfaction, la situation a peut être préférée à la situation b sur les critères i et j ou vice versa. L'exemple suivant illustre cette forme de dépendance dans notre contexte.

Tableau 4.3 – Exemple de niveaux de qualité illustrant les dépendances préférentielles

	Temps de réponse	**Fiabilité**	**Disponibilité**
$QoSL_1(orch)$	0.7	0.95	0.9
$QoSL_2(orch)$	0.9	0.95	0.7
$QoSL_3(orch)$	0.9	0.5	0.7
$QoSL_4(orch)$	0.7	0.5	0.9

Exemple : considérons quatre niveaux de qualité $QoSL_1(orch), \ldots, QoSL_4(orch)$ d'une orchestration de services impliquant trois attributs qualité (critères) qui sont le temps de réponse, la fiabilité et la disponibilité. Ces niveaux de qualité sont présentés dans le tableau 4.3. Les valeurs des attributs qualité données sont déjà normalisées et représentent donc la satisfaction de chacun des attributs qualité. Le client est interrogé sur son rangement de ces niveaux de qualité. Il est clair que $QoSL_1(orch) \succ QoSL_4(orch)$ et $QoSL_2(orch) \succ QoSL_3(orch)$. Par contre, la comparaison de $QoSL_1(orch)$ et $QoSL_2(orch)$ d'une part, et de $QoSL_3(orch)$ et $QoSL_4(orch)$ d'autre part, n'est pas évidente puisque les satisfactions des attributs qualité sont croisées. Le client peut avoir le raisonnement suivant : si la fiabilité est bonne, il préfère la disponibilité sur le temps de réponse ($QoSL_1(orch) \succ QoSL_2(orch)$), alors que si la fiabilité est mauvaise, il préfère le temps de réponse à la disponibilité ($QoSL_3(orch) \succ QoSL_4(orch)$). Ainsi, le temps de réponse et la disponibilité sont préférentiellement dépendants de la fiabilité.

Dans notre approche, nous considérons uniquement les dépendances préférentielles. En outre, nous supposons que les attributs qualité, impliqués dans la mesure du degré de satisfaction, ne sont ni interchangeables ni complémentaires. En effet, cette hypothèse fait référence à la propriété de *non redondance* qui est une des propriétés que l'ensemble des critères doit respecter afin que les critères soient cohérents [Henriet, 2000]. Si deux attributs qualité i et j sont interchangeables (comme la fiabilité et la précision), nous ne gardons qu'un seul attribut qualité. D'autre part, si deux attributs qualité sont complémentaires (comme le temps d'exécution et la latence), nous les remplaçons par un seul attribut qualité de plus haut niveau (par exemple le temps de réponse).

Nous nous intéressons donc dans la suite aux dépendances préférentielles que peut présenter le client (décideur). Ces dépendances ne peuvent pas être modélisées par les opérateurs d'agrégation additifs tels que la moyenne pondérée [Marichal, 2000, Grabisch and Labreuche, 2008]. En effet, reprenons l'exemple précédent et les préférences exprimées par le client. Ces préférences peuvent être traduites par les deux inégalités suivantes :

$$QoSL_1(orch) \succ QoSL_2(orch) \Longleftrightarrow F((0.7, 0.95, 0.9)) > F((0.9, 0.95, 0.7)) \quad (4.9)$$

$$QoSL_3(orch) \succ QoSL_4(orch) \Longleftrightarrow F((0.9, 0.5, 0.7)) > F((0.7, 0.5, 0.9)) \quad (4.10)$$

où F est la fonction d'utilité globale. Supposons que F soit la moyenne pondérée (voir équation 4.7), et, w_1,w_2 et w_3 désignent respectivement les poids d'importance relatifs au temps de réponse, à la fiabilité et à la disponibilité. L'inégalité 4.9 implique que la disponibilité est plus importante que le temps de réponse, c'est-à-dire $w_3 > w_1$. D'autre part, l'inégalité 4.10 traduit exactement le contraire (c'est-à-dire $w_3 < w_1$), ce qui est absurde. Par conséquent, nous pouvons conclure que la moyenne arithmétique pondérée ne peut pas prendre en compte les dépendances préférentielles. Ceci nous a conduit à chercher d'autres méthodes/opérateurs d'agrégation qui permettent de répondre à notre objectif, la prise en compte des dépendances préférentielles. Dans ce sens, nous avons étudié

un opérateur d'agrégation qui généralise la moyenne pondérée et qui permet de prendre en compte les dépendances préférentielles entre les attributs qualité, c'est l'intégrale de Choquet. Nous allons nous intéresser dans la suite à cet opérateur d'agrégation et nous allons montrer comment il rend possible la prise en compte des dépendances préférentielles dans la construction de la fonction d'utilité globale et par conséquent dans le modèle de préférences.

4.4.2 Étude de l'intégrale de Choquet

Afin de couvrir les limites de l'opérateur d'agrégation de la moyenne pondérée, une solution consiste à attribuer des poids d'importance sur des coalitions d'attributs qualité (groupes ou sous-ensembles d'attributs qualité) [Grabisch and Labreuche, 2005]. Ceci peut être réalisé en introduisant une fonction particulière, définie sur l'ensemble de tous les sous-ensembles d'attributs qualité, appelée *mesure floue* ou *capacité* [Marichal, 2002, Grabisch and Labreuche, 2005]. Soit $P = \{1, \ldots, p\}$ l'ensemble des attributs qualité, $\mathcal{P}(P)$ est l'ensemble de tous les sous-ensembles de P, avec $|\mathcal{P}(P)| = 2^p$.

Définition 4.2. [Marichal, 2002, Grabisch and Labreuche, 2005]. Une mesure floue ou une capacité est une fonction $\mu : \mathcal{P}(P) \longrightarrow [0, 1]$ vérifiant :

- $A \subset B \Rightarrow \mu(A) \leq \mu(B)$, où A et B sont deux sous-ensembles de P. Cette propriété représente la monotonie de la capacité. Elle signifie que le poids d'importance d'un groupe d'attributs qualité augmente lorsque le nombre d'attributs qualité augmente.
- $\mu(\emptyset) = 0$
- $\mu(P) = 1$ (capacité normalisée)

Notons que la fonction μ (capacité) n'est pas une donnée qui définit les valeurs $\mu(A)$ des poids d'importance d'un ensemble A d'attributs qualité, $\forall\ A \subset P$. Ces valeurs doivent être construites. Ainsi, pour définir entièrement la capacité μ, il faut déterminer 2^p valeurs $\mu(A)$ pour tous les sous-ensembles possibles A de P. Une fois que ces capacités sont construites, il convient de les agréger avec les satisfactions (valeurs normalisées) de chaque attribut qualité $u_i(q_i)$, $i \in P$, pour déterminer le degré de satisfaction du niveau de qualité correspondant. Un opérateur d'agrégation convenable, qui généralise l'opérateur de la moyenne pondérée, est l'intégrale de Choquet [Marichal, 2002, Grabisch and Labreuche, 2005] défini comme suit :

Définition 4.3. Soit $x := (x_1, \ldots, x_p) \in \mathbb{R}^p$ et μ une capacité sur P. L'intégrale de Choquet (discrète) de x par rapport à μ est définie par :

$$F_\mu(x) := \sum_{i=1}^{p} x_{\sigma(i)}[\mu(A_{\sigma(i)}) - \mu(A_{\sigma(i+1)})] \tag{4.11}$$

où $\sigma(\ )$ indique une permutation sur P tel que $x_{\sigma(1)} \leq \cdots \leq x_{\sigma(p)}$ et $A_{\sigma(i)} = \{\sigma(i), \ldots, \sigma(p)\}$, avec $A_{\sigma(p+1)} = \emptyset$.

Exemple : soit $x := (x_1, x_2, x_3)$ tel que $x_3 \leq x_1 \leq x_2$. Nous avons donc : $x_{\sigma(1)} = x_3$, $x_{\sigma(2)} = x_1$, $x_{\sigma(3)} = x_2$, $A_{\sigma(1)} = \{3, 1, 2\}$, $A_{\sigma(2)} = \{1, 2\}$ et $A_{\sigma(3)} = \{2\}$. De fait :

$$\begin{aligned} F_\mu(x) &= x_{\sigma(1)}[\mu(A_{\sigma(1))} - \mu(A_{\sigma(2)})] + x_{\sigma(2)}[\mu(A_{\sigma(2)}) - \mu(A_{\sigma(3)})] + x_{\sigma(3)}\mu(A_{\sigma(3)}) \\ &= x_3[\mu(\{3,1,2\}) - \mu(\{1,2\})] + x_1[\mu(\{1,2\}) - \mu(\{2\})] + x_2\mu(2) \ . \end{aligned}$$

Cependant, le fait que la capacité μ soit définie sur l'ensemble $\mathcal{P}(P)$ des sous-ensembles, rend le problème exponentiellement complexe (2^p valeurs sont requises pour définir la capacité μ) et difficile à traiter pour un ensemble large d'attributs qualité [Grabisch and Labreuche, 2008, Mayag et al., 2011]. En outre, il est difficile d'interpréter les valeurs de la capacité et par suite d'analyser le comportement de l'intégrale de Choquet [Grabisch and Labreuche, 2008]. Pour ces raisons, plusieurs modèles, plus simples, ont été proposés tels que les capacités symétriques ainsi que leur généralisation appelée les capacités p-symétriques [Miranda et al., 2002], les capacités k-additives [Grabisch, 1997], etc. Un cas particulièrement intéressant est la capacité 2-additive qui permet de représenter et d'interpréter toute interaction (dépendance) entre deux critères, les interactions plus complexes (c'est-à-dire impliquant plus que deux critères) sont dans ce cas négligées. [Grabisch, 2006] mentionne qu'on démontre expérimentalement que l'on gagne peu en précision du modèle en passant d'une capacité 2-additive à une capacité k-additive. Par contre, on perd beaucoup en passant d'une capacité 2-additive à une capacité 1-additive dont l'intégrale de Choquet correspondante n'est dans ce cas qu'une moyenne pondérée. Ceci implique dans notre contexte, qu'un modèle de préférences basé sur la moyenne pondérée est peu précis par rapport à un autre modèle basé sur une capacité 2-additive par exemple. En outre, il est difficile d'appréhender le sens des interactions au delà de deux critères par un décideur humain [Grabisch, 2006]. L'intégrale de Choquet 2-additive représente donc un bon compromis entre la précision et la complexité du modèle [Mayag et al., 2011], elle nécessite seulement ($\sum_{i=1}^{2} C_p^i = p(p+1)/2$) paramètres à déterminer au lieu de 2^p dans le cas de sa forme la plus générale.

L'intégrale de Choquet par rapport à une capacité 2-additive, que nous appelons par la suite l'intégrale de Choquet 2-additive plus simplement, a été largement utilisée dans diverses applications telles que l'évaluation de l'inconfort en position assise [Grabisch et al., 2002], la construction d'un système d'indicateurs de performance pour l'expression de la performance industrielle [Clivillé et al., 2007, Berrah and Clivillé, 2007, Berrah et al., 2008] et la conception des systèmes complexes [Pignon and Labreuche, 2007]. Dans nos travaux, nous appliquons l'intégrale de Choquet 2-additive dans le domaine des architectures orientées services. Nous allons considérer cet opérateur dans la construction du modèle de préférences du client afin qu'il soit le plus conforme possible aux préférences réelles du client. Rappelons que le modèle de préférences, qui fait intervenir l'opérateur de l'intégrale de Choquet 2-additive, est utilisé pour déterminer le degré de satisfaction des orchestrations de services. Ce dernier constitue l'information de base que le système de supervision est chargé de surveiller. C'est pour cette raison que nous avons besoin de la précision de

peur que le système de supervision ne puisse pas réaliser sa fonction convenablement (soit il mène des actions inutiles, soit au contraire, il reste neutre alors qu'il faudrait agir).

L'intégrale de Choquet 2-additive est un opérateur d'agrégation qui est défini par le biais des $(p(p+1)/2)$ paramètres. Par conséquent, avant d'appliquer cet opérateur, il faut tout d'abord déterminer ces paramètres de telle sorte que le modèle de préférences représente au mieux les préférences du client (décideur). Pour ce faire, nous proposons d'utiliser la méthode MACBETH (voir section 4.3) qui permet de déterminer les fonctions d'utilité marginales et la fonction d'utilité globale (les paramètres de l'opérateur d'agrégation) d'une manière cohérente, respectant la théorie de mesurage [Clivillé, 2004, Labreuche and Grabisch, 2003]. Néanmoins, la méthode MACBETH est basée sur l'opérateur d'agrégation de la moyenne pondérée, un opérateur qui ne permet pas de prendre en compte les dépendances entre critères (voir section 4.4.1). Plusieurs travaux [Labreuche and Grabisch, 2003, Clivillé, 2004, Berrah and Clivillé, 2007, Grabisch and Labreuche, 2008, Mayag et al., 2011] ont été conduits dans le but d'étendre la méthode MACBETH de manière à prendre en compte les interactions entre les critères.

Pour toutes les raisons développées précédemment, nous avons choisi d'appliquer la méthode MACBETH étendue à l'intégrale de Choquet 2-additive afin de prendre en compte les dépendances préférentielles dans la mesure du degré de satisfaction [Fakhfakh et al., 2011b, Fakhfakh et al., 2011d, Fakhfakh et al., 2011e].

4.4.3 Application de la méthode MACBETH étendue avec l'intégrale de Choquet 2-additive

Comme nous l'avons présenté dans la section 4.3, la méthode MACBETH comporte principalement quatre étapes à savoir (voir figure 4.1) :

- la définition du contexte ;
- la construction des fonctions d'utilité marginales (normalisation) ;
- la construction de la fonction d'utilité globale ;
- l'agrégation.

Le contexte étant toujours le même, l'étape de construction des fonctions d'utilité marginales, quant à elle, reste inchangée. En effet, la dépendance entre les attributs qualité n'intervient pas dans cette étape, puisque dans la démarche de construction des valeurs normalisées, chaque attribut qualité est traité séparément. Autrement dit, dans la procédure d'expression des préférences, le client ne prend pas en compte les valeurs des autres attributs qualité (il peut imaginer par exemple que les autres valeurs d'attributs qualité sont les mêmes pour tous les niveaux de qualité). Par contre, l'étape de construction de la fonction d'utilité globale (la détermination des paramètres de l'opérateur d'agrégation) et l'étape d'agrégation nécessitent d'être adaptées. Ceci est dû à ce que l'opérateur d'agrégation n'est plus le même et implique plus de paramètres que la moyenne pondérée, $(p(p+1)/2)$ paramètres au lieu de p paramètres (poids d'importance) pour la moyenne pondérée.

Étant donné un niveau de qualité $QoSL(orch) = (q_1, \ldots, q_p)$ et ayant construit les fonctions d'utilité marginales u_i, $i \in P$, l'intégrale de Choquet 2-additive peut être exprimée directement à l'aide des poids d'importance et des indices d'interaction [Berrah and Clivillé, 2007, Grabisch and Labreuche, 2008, Mayag et al., 2011] :

$$F_\mu\Big(u_1(q_1), ..., u_p(q_p)\Big) = \sum_{i=1}^{p} \nu_i.u_i(q_i) - \frac{1}{2} \sum_{\substack{\{i,j\}\in P \\ i\neq j}} I_{ij}|u_i(q_i) - u_j(q_j)| \ . \tag{4.12}$$

Cette expression fait intervenir deux types de paramètres :

- les indices de Shapley ν_i (de la capacité μ) qui représentent les poids d'importance de chaque attribut qualité i, avec $\sum_{i=1}^{p} \nu_i = 1$,
- les paramètres d'interaction I_{ij} qui quantifient les interactions mutuelles entre deux attributs qualité i et j, avec $I_{ij} \in [-1, 1]$. Les paramètres d'interaction (I_{ij}) peuvent être:
  - positifs, ce qui implique qu'il y a une contradiction entre les attributs qualité i et j. En effet, plus $u_i(q_i)$ est différent de $u_j(q_j)$, plus le phénomène d'interaction pénalise le degré de satisfaction. Ceci correspond au cas de l'auto-affaiblissement où la contribution conjointe des attributs qualité i et j est moins significative que la somme des contributions de chacun des attributs qualité $(\mu_{ij} < \mu_i + \mu_j)$;
  - négatifs, ce qui implique une synergie positive entre les attributs qualité i et j. En effet, plus $u_i(q_i)$ est différent de $u_j(q_j)$, plus le phénomène d'interaction augmente le degré de satisfaction. Ceci correspond au cas de l'auto-renforcement où la contribution conjointe des attributs qualité i et j est plus significative que la somme des contributions de chacun des attributs qualité $(\mu_{ij} > \mu_i + \mu_j)$;
  - nuls, ce qui implique que les critères i et j sont indépendants. Si tous les paramètres d'interactions sont nuls, l'intégrale de Choquet 2-additive devient équivalent à la moyenne pondérée.

Les indices de Shapley ν_i doivent respecter la condition de monotonie suivante :

$$\forall\, i \in P,\ j \neq i, \quad \nu_i - \frac{1}{2} \sum_{j=1}^{p} |I_{ij}| \geq 0 \ . \tag{4.13}$$

Voyons maintenant comment déterminer les paramètres qui interviennent dans l'intégrale de Choquet 2-additive.

Construction de la fonction d'utilité globale basée sur l'intégrale de Choquet 2-additive

Nous avons vu précédemment que l'intégrale de Choquet 2-additive est entièrement définie par ses $(p(p+1)/2)$ paramètres. Afin de déterminer ces paramètres par la méthode MACBETH étendue, nous avons besoin d'autant d'intensités de préférences, exprimées par le client, que le nombre de paramètres (voir section 4.3.3). De la même manière que

Tableau 4.4 – Préférences et intensités de préférence pour la détermination des paramètres de l'intégrale de Choquet 2-additive

	$b_{\sigma(1)} = 1_P$	$b_{\sigma(2)}$	...	$b_{\sigma(\lvert\mathbb{B}\rvert-1)}$	$b_{\sigma(\lvert\mathbb{B}\rvert)} = 0_P$
$b_{\sigma(1)} = 1_P$	nulle	h_1	...	P	P
$b_{\sigma(2)}$		nulle	...	P	P
$\vdots$			nulle	...	...
$b_{\sigma(\lvert\mathbb{B}\rvert-1)}$				nulle	$h_{\lvert\mathbb{B}\rvert-1}$
$b_{\sigma(\lvert\mathbb{B}\rvert)} = 0_P$					nulle

$h_i \in \{0 = nulle,\ 1 = très\, faible,\ 2 = faible,\ 3 = modérée,\ 4 = fort,\ 5 = très\, fort,\ 6 = extrême\}$

dans le cas de la moyenne pondérée, nous allons considérer les niveaux de qualité binaires. Nous rappelons que dans le cas de la moyenne pondérée, nous avons utilisé des niveaux de qualité binaires du type $(1_i, 0_{-i})$ où seulement l'attribut qualité i est totalement satisfait (égal à q_i^{good}). Cependant, ceci ne permet pas d'exprimer le poids d'importance sur les coalitions d'attributs qualité. Pour cette raison, nous considérons également les niveaux de qualité binaires du type $(1_{ij}, 0_{-ij})$ [Labreuche and Grabisch, 2003, Berrah and Clivillé, 2007, Grabisch and Labreuche, 2008, Mayag et al., 2011], où les attributs qualité i et j sont totalement satisfaits (égaux à q_i^{good} et q_j^{good}) et les autres sont au minimum acceptables. Notons par $\mathbb{B}$ l'ensemble des niveaux de qualité binaires :

$$\mathbb{B} := \{0_P, (1_i, 0_{-i}), (1_{ij}, 0_{-ij}), 1_P; i, j \in P, i \neq j\} \ .$$

où $0_P := (0, \ldots, 0)$ et $1_P := (1, \ldots, 1)$. Dans ce cas, l'expression de l'intégrale de Choquet d'un niveau de qualité binaire peut s'écrire sous la forme :

$$F_\mu(0_P) = \mu_\emptyset = 0 \tag{4.14}$$

$$F_\mu\Big((1_i, 0_{-i})\Big) = \mu_i = \nu_i - \frac{1}{2} \sum_{k \in P, k \neq i} I_{ik} \tag{4.15}$$

$$F_\mu\Big((1_{ij}, 0_{-ij})\Big) = \mu_{ij} = \nu_i + \nu_j - \frac{1}{2} \sum_{k \in P, k \notin \{i,j\}} (I_{ik} + I_{jk}) \tag{4.16}$$

$$F_\mu(1_P) = \mu_P = \sum_{i=1}^{p} \nu_i = 1 \ . \tag{4.17}$$

En suivant la démarche de la méthode MACBETH, le client est amené, tout d'abord, à ranger les niveaux de qualité binaires, $b_i \in \mathbb{B}$, par ordre de préférence. Ceci revient à ranger les valeurs de la capacité 2-additive μ par ordre de préférence. Ensuite, le client est interrogé sur ses intensités de préférences $h \in \{0, \ldots, 6\}$, exprimées sur la différence d'attractivité $F_\mu(b_i) - F_\mu(b_j)$ entre deux niveaux de qualité binaires b_i et b_j, tel que $b_i, b_j \in \mathbb{B}$ et $F_\mu(b_i) \succ F_\mu(b_j)$. Ces informations sont traduites dans la matrice donnée par le tableau 4.4. Nous rappelons que la matrice est remplie de la même manière que celles données dans

les tableaux 4.1 et 4.2 (voir section 4.3). Ceci nous fournit $[p(p+1)/2+1]$ équations dont chacune est de la forme :

$$F_\mu(b_i) - F_\mu(b_j) = h\alpha \ . \tag{4.18}$$

En remplaçant $F_\mu(b_i)$ et $F_\mu(b_j)$ de l'équation 4.18 par leur expression correspondante parmi les équations 4.14, 4.15, 4.16 et 4.17, nous obtenons un système d'équations linéaires qui peut se mettre sous la forme $A \times X = B$. La résolution de système d'équations permet ainsi de déterminer les paramètres de l'intégrale de Choquet 2-additive. Notons que les indices de Shapley ν_i doivent respecter la condition de monotonie 4.13. Généralement, cette condition de monotonie est respectée, en particulier dans le cas de trois attributs qualité. Cependant, le non respect par la solution obtenue de cette condition de monotonie, implique potentiellement une incohérence dans l'expression des intensités de préférences auquel cas les intensités de préférences exprimées doivent être reconsidérées.

Exemple : considérons les trois attributs qualité suivants : temps de réponse (q_{rt}), fiabilité (q_{rel}) et disponibilité (q_{av}). Ces trois attributs qualité forment le niveau de qualité $(q_{rt}, q_{rel}, q_{av})$, noté plus simplement (q_1, q_2, q_3). Nous avons donc $(3\times4/2 = 6)$ paramètres à déterminer pour définir l'intégrale de Choquet 2-additive : trois indices de Shapley ν_1, ν_2, ν_3 et trois paramètres d'interaction I_{12}, I_{13}, I_{23}. L'ensemble $\mathbb{B}$ des niveaux de qualité binaires est dans ce cas :

$$\mathbb{B} = \{(0,0,0), (1,0,0), (0,1,0), (0,0,1), (1,1,0), (1,0,1), (0,1,1), (1,1,1)\} \ .$$

Le client est donc sollicité pour classer ces situations par ordre de préférence. Supposons qu'il effectue le classement suivant :

$$F_\mu\Big((1,1,1)\Big) \succ F_\mu\Big((0,1,1)\Big) \succ F_\mu\Big((1,1,0)\Big) \succ F_\mu\Big((0,1,0)\Big) \succ F_\mu\Big((1,0,1)\Big) \succ F_\mu\Big((0,0,1)\Big) \succ F_\mu\Big((1,0,0)\Big) \succ F_\mu\Big((0,0,0)\Big) \ .$$

Par suite, le client exprime ses intensités de préférence $h_1, \ldots, h_7$ de la façon décrite par la matrice donnée par le tableau 4.5. Nous pouvons donc déduire l'ensemble des équations suivantes :

$$\begin{aligned}
F_\mu\Big((1,1,1)\Big) - F_\mu\Big((0,1,1)\Big) &= h_1\alpha = \nu_1 + 0.5I_{12} + 0.5I_{13} \\
F_\mu\Big((0,1,1)\Big) - F_\mu\Big((1,1,0)\Big) &= h_2\alpha = -\nu_1 + \nu_3 - 0.5I_{12} + 0.5I_{23} \\
F_\mu\Big((1,1,0)\Big) - F_\mu\Big((0,1,0)\Big) &= h_3\alpha = \nu_1 + 0.5I_{12} - 0.5I_{13} \\
F_\mu\Big((0,1,0)\Big) - F_\mu\Big((1,0,1)\Big) &= h_4\alpha = -\nu_1 + \nu_2 - \nu_3
\end{aligned}$$

Tableau 4.5 – Matrice de préférence dans le cas de trois attributs qualité

	$(1,1,1)$	$(0,1,1)$	$(1,1,0)$	$(0,1,0)$	$(1,0,1)$	$(1,0,0)$	$(0,0,1)$	$(0,0,0)$
$(1,1,1)$	No	h_1	P	P	P	P	P	P
$(0,1,1)$		No	h_2	P	P	P	P	P
$(1,1,0)$			No	h_3	P	P	P	P
$(0,1,0)$				No	h_4	P	P	P
$(1,0,1)$					No	h_5	P	P
$(1,0,0)$						No	h_6	P
$(0,0,1)$							No	h_7

$$
\begin{aligned}
F_\mu\Big((1,0,1)\Big) - F_\mu\Big((1,0,0)\Big) &= h_5\alpha = \nu_3 + 0.5I_{13} - 0.5I_{23} \\
F_\mu\Big((1,0,0)\Big) - F_\mu\Big((0,0,1)\Big) &= h_6\alpha = \nu_1 - \nu_3 - 0.5I_{12} + 0.5I_{23} \\
F_\mu\Big((0,0,1)\Big) - F_\mu\Big((0,0,0)\Big) &= h_7\alpha = \nu_3 - 0.5I_{13} - 0.5I_{23} \ .
\end{aligned}
$$

Sachant que les indices de Shapley ν_i doivent vérifier la condition $\nu_1 + \nu_2 + \nu_3 = 1$, le système d'équations peut se mettre sous la forme matricielle $A \times X = B$ suivante :

$$
\underbrace{\begin{bmatrix}
1 & 0 & 0 & 0.5 & 0.5 & 0 & -h_1 \\
-1 & 0 & 1 & -0.5 & 0 & 0.5 & -h_2 \\
1 & 0 & 0 & 0.5 & -0.5 & 0 & -h_3 \\
-1 & 1 & -1 & 0 & 0 & 0 & -h_4 \\
0 & 0 & 1 & 0 & 0.5 & -0.5 & -h_5 \\
1 & 0 & -1 & -0.5 & 0 & 0.5 & -h_6 \\
0 & 0 & 1 & 0 & -0.5 & -0.5 & -h_7 \\
1 & 1 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}}_{A} \times \underbrace{\begin{bmatrix}
\nu_1 \\ \nu_2 \\ \nu_3 \\ I_{12} \\ I_{13} \\ I_{23} \\ \alpha
\end{bmatrix}}_{X} = \underbrace{\begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1
\end{bmatrix}}_{B} \tag{4.19}
$$

où $h_1, \ldots, h_7$ de la matrice A sont des données. En résolvant ce système d'équations, l'intégrale de Choquet 2-additive est ainsi entièrement définie.

Agrégation

Après avoir déterminé les paramètres (ν_i, I_{ij}) de l'intégrale de Choquet 2-additive, la valeur du *degré de satisfaction* est obtenue en appliquant l'expression de l'intégrale de Choquet 2-additive donnée en équation 4.12.

La méthode présentée dans cette section permet non seulement de prendre en compte les préférences du client et son importance accordée aux attributs qualité, mais aussi les dépendances préférentielles qui peuvent exister. L'exemple suivant illustre la prise en compte des dépendances préférentielles du client.

Exemple : reprenons l'exemple des dépendances préférentielles présentée dans la section 4.4.1. Rappelons que les préférences du client concernant les quatre niveaux de qualité sont traduites par la relation suivante :

$$QoSL_1(orch) \succ QoSL_2(orch) \succ QoSL_3(orch) \succ QoSL_4(orch)$$

Nous avons montré dans la section 4.4.1 que ces préférences ne peuvent pas être modélisées par l'opérateur de la moyenne pondérée. Considérons par exemple le modèle de l'intégrale de Choquet 2-additive défini par [Fakhfakh et al., 2011d] :

$$\nu_1 = 0.22,\ \nu_2 = 0.54,\ \nu_3 = 0.24$$

$$I_{12} = -0.13,\ I_{13} = 0.04,\ I_{23} = 0.26$$

En agrégeant les niveaux de qualité données en tableau 4.3 par cette fonction d'utilité globale, nous pouvons vérifier que les degrés de satisfaction relatives aux quatre niveaux de qualité satisfont bien les préférences du client :

$$F_\mu(QoSL_1) = 0.89 > F_\mu(QoSL_2) = 0.85 > F_\mu(QoSL_3) = 0.51 > F_\mu(QoSL_4) = 0.47$$

Cet exemple montre que le degré de satisfaction peut être utilisé pour comparer, par exemple, plusieurs instances (exécutions) d'une même orchestration ou encore plusieurs modèles d'orchestration supportant le même processus métier. En effet, imaginons que chaque niveau de qualité et par suite chaque degré de satisfaction correspond à une instance d'orchestration (respectivement à un modèle d'orchestration), la comparaison des degrés de satisfaction correspondant permet ainsi de comparer et d'analyser la qualité des instances d'une même orchestration de services (respectivement des modèles d'orchestration supportant un même processus métier).

Synthèse

Nous avons abordé dans cette section les différentes formes de dépendances entre les attributs qualité et montré que la méthode MACBETH (basée sur la moyenne pondérée) ne permet pas de prendre en compte ces dépendances, en particulier les dépendances préférentielles. Pour cela, nous nous sommes orientés vers des opérateurs d'agrégation permettant la prise en compte des dépendances entre les attributs qualité. Nous avons étudié un opérateur qui généralise la moyenne pondérée qui est l'intégrale de Choquet. Cet opérateur permet de prendre en compte les dépendances/interactions entre les attributs qualité par l'affectation des poids sur des coalitions d'attributs qualité. Cependant, il s'est avéré que l'intégrale de Choquet, sous sa forme générale, nécessite 2^p paramètres et donc sa construction est relativement complexe, notamment si nous considérons sa construction par l'expression de préférences par le client. Notre choix a porté sur l'intégrale de Choquet 2-additive, un cas plus simple et plus intéressant car il représente un compromis entre

complexité ($p(p+1)/2$ paramètres au lieu de 2^p) et représentation des dépendances entre les attributs qualité. Nous avons donc choisi cet opérateur d'agrégation combiné à la méthode MACBETH pour la définition du modèle de préférences du client. Dans ce sens, nous avons montré à travers un exemple la prise en compte dans l'évaluation du degré de satisfaction des dépendances préférentielles que peut manifester le client. Nous avons également décrit la procédure pour construire la fonction d'utilité globale basée sur l'intégrale de Choquet 2-additive en utilisant les préférences et les intensités de préférences exprimées par le client. Ainsi, le modèle de préférences (fonctions d'utilité marginales et fonction d'utilité globale) est construit uniquement à partir des informations fournies par le client. Pour cette raison et du fait de la prise en compte des dépendances préférentielles, nous estimons que notre modèle de préférences permet de mieux représenter la satisfaction réelle du client.

Par ailleurs, notre principal objectif est la préservation de la satisfaction des exigences du client tout au long de l'exécution. Pour ce faire, il convient d'évaluer le degré de satisfaction tout au long de l'exécution et de préciser les stratégies de surveillance et les scénarios d'adaptation en cas de besoin ; c'est le sujet de la section suivante.

4.5 Gestion du degré de satisfaction des orchestrations

Cette section est consacrée au contrôle et à la gestion dynamique des orchestrations de services. À ce stade (voir figure 4.4), nous avons présenté l'approche globale d'agrégation incluant l'agrégation par les règles de patrons de *workflow* et l'agrégation basée sur une méthode d'aide à la décision multi-critères. Cette dernière, permettant de construire le modèle de préférences du client, fournit le degré de satisfaction de l'orchestration. L'objectif dans cette section, est de définir des stratégies de surveillance pour le contrôle des orchestrations de services au cours de l'exécution. Pour chacune des stratégies de surveillance, nous allons spécifier la condition de déclenchement de l'adaptation. Ces conditions de déclenchement, lorsqu'elles sont vérifiées par le déclencheur (voir figure 4.4), signifie qu'une dégradation de la qualité globale de l'orchestration est survenue. Ainsi, un ou plusieurs scénarios d'évolution, que nous allons proposer, doivent être mis en place afin de rétablir la qualité de l'orchestration. Rappelons que le choix d'un ou des scénarios d'évolution adéquats (décision) sera le sujet de nos travaux futurs. Avant de procéder à la concrétisation de ces différents objectifs, nous allons revenir à la détermination du degré de satisfaction de l'orchestration.

Nous avons vu, dans le chapitre précédent, la méthode d'agrégation par les règles de patrons de *workflow* durant les différentes phases du cycle de vie de l'orchestration, à savoir pendant la phase de conception, durant l'exécution et à la fin de l'exécution. Il convient, de la même manière, de déterminer le degré de satisfaction pour chacune de ces phases. Ainsi, par cohérence, le degré de satisfaction de l'orchestration pendant les deux premières phases est une variable aléatoire tandis qu'à la fin de l'exécution, le degré de satisfaction est une valeur déterministe. Le modèle de préférences étant défini, l'agrégation des valeurs déterministes se fait par application directe de la méthode présentée dans ce chapitre

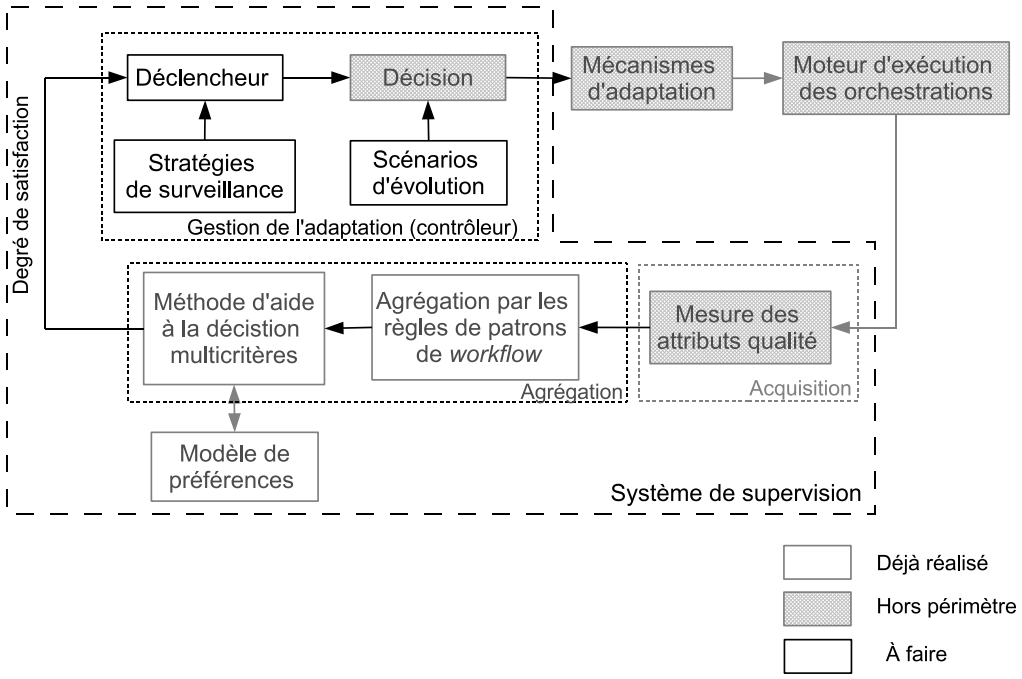


Figure 4.4 – Bilan des objectifs de nos travaux

(voir sections 4.3 et 4.4.3). Par contre, lors des phases de conception et d'exécution, il est nécessaire de réaliser un traitement différent.

4.5.1 Détermination de la distribution du degré de satisfaction

Pendant la phase de conception et au cours de l'exécution de l'orchestration, nous avons à disposition dans le cas général, des variables aléatoires discrètes $X_1, \ldots, X_p$ représentant les attributs qualité i, $i \in P$. Ces variables aléatoires sont issues du résultat de l'agrégation par les règles de patrons de *workflow* (voir sections 3.4 et 3.5). Notons que les valeurs déterministes sont un cas particulier des variables aléatoires discrètes où leur domaine est limité à la valeur elle-même et leur loi de probabilité est mise à 1. Pour cela, nous présentons la détermination de la distribution de probabilité du degré de satisfaction dans le cas général puis nous la spécialiserons pour le cas des trois attributs qualité que nous avons retenus.

La distribution du degré de satisfaction est obtenue en agrégeant les variables aléatoires de chacun des attributs qualité par la méthode d'aide à la décision présentée dans ce chapitre. Notons par DS la variable aléatoire désignant le degré de satisfaction. Nous supposons ici que les fonctions d'utilité marginales u_i et la fonction d'utilité globale (l'intégrale de Choquet 2-additive) ont été déjà construites auparavant. La détermination de la variable aléatoire $DS = F_\mu\Big((u_1(X_1), \ldots, u_p(X_p)\Big)$ consiste à déterminer son domaine $Dom(Z)$ et sa loi de probabilité P_Z.

Détermination du domaine de DS ($Dom(DS)$) : le domaine de DS est $Dom(DS) = \{z_k | z_k = F_\mu\Big((u_1(x_1), \ldots, u_p(x_p)\Big), \forall\ x_i \in Dom(X_i)\}$. Autrement dit, pour chaque variable aléatoire X_i, $i \in P$, nous effectuons la normalisation de chaque valeur de $Dom(X_i)$ à l'aide de la fonction d'utilité marginale u_i. Ensuite, nous déterminons toutes les combinaisons possibles des niveaux de qualité. Dans ce cas, nous avons $\prod_{i=1}^{p} |Dom(X_i)|$ niveaux de qualité. Enfin, nous calculons les degrés de satisfaction z_k correspondant à ces niveaux de qualité.

Dans le cas des trois attributs qualité retenus, le temps de réponse est représenté par la variable aléatoire X_1 dont le domaine est $Dom(X_1) = \{x_1, \ldots, x_n\}$, la fiabilité $X_2 = q_{rel}$ et la disponibilité $X_3 = q_{av}$. Par conséquent, le domaine du degré de satisfaction $Dom(DS)$ est constitué de l'ensemble des degrés de satisfaction relatifs aux triplets (x_i, q_{rel}, q_{av}), pour tout $x_i \in Dom(X_i)$:

$$Dom(DS) = \{z_k | z_k = F_\mu\left(u_1(x_i), u_2(q_{rel}), u_3(q_{av})\right),\ \forall x_i \in Dom(X_1)\}\ . \tag{4.20}$$

Détermination de la loi de probabilité de DS (P_{DS}) : la probabilité de $z_k \in Dom(DS)$ est calculée de la manière suivante :

$$P_{DS}(z_k) = \sum_{\substack{z_k = F_\mu(u_1(x_1), \ldots, u_p(x_p)) \\ x_i \in Dom(X_i) \\ 1 \leq i \leq p}} P_{X_1 \ldots X_p}(x_1, \ldots, x_p)$$

où $P_{X_1 \ldots X_p}(x_1, \ldots, x_p)$ est la loi de probabilité jointe des variables aléatoires discrètes $X_1, \ldots, X_p$. Elle est déterminée, dans le cas général, par l'expression suivante :

$$\begin{aligned} P_{X_1 \ldots X_p}(x_1, \ldots, x_p) &= P_{X_p | X_1 \ldots X_{p-1}}(x_p | x_1, \ldots, x_{p-1}) \times P_{X_1 \ldots X_{p-1}}(x_1, \ldots, x_{p-1}) \\ &= P_{X_p | X_1 \ldots X_{p-1}}(x_p | x_1, \ldots, x_{p-1}) \times P_{X_{p-1} | X_1 \ldots X_{p-2}}(x_{p-1} | x_1, \ldots, x_{p-2}) \\ &= \ldots \\ &= P_{X_p | X_1 \ldots X_{p-1}}(x_p | x_1, \ldots, x_{p-1}) \times \cdots \times P_{X_2 | X_1}(x_2 | x_1).P_{X_1}(x_1) \end{aligned} \tag{4.21}$$

où $P_{X|Y}(x|y)$ est la loi de probabilité conditionnelle de X sachant $Y = y$. Notons que la probabilité :

$$P_{XY}(x, y) = P_{X|Y}(x|y).P_Y(y) \tag{4.22}$$

$$= P_{Y|X}(y|x).P_X(x)\ . \tag{4.23}$$

Ceci implique que la loi de probabilité jointe de plusieurs variables aléatoires peut être exprimée à l'aide des probabilités conditionnelles selon la connaissance que l'on a, *a priori*, sur ces probabilités conditionnelles. Par exemple, dans le cas de la probabilité jointe P_{XY} de deux variables aléatoires X et Y, nous pouvons utiliser l'expression (4.22) si nous connaissons $P_{X|Y}(x|y)$ ou l'expression (4.23) si nous savons calculer $P_{Y|X}(y|x)$. Cette

question ne se pose pas lorsque les variables aléatoires X_i, $i \in P$, sont indépendantes. En effet, dans ce cas, l'équation 4.21 devient :

$$P_{X_1 \ldots X_p}(x_1, \ldots, x_p) = \prod_{i=1}^{p} P_{X_i}(x_i) \ . \tag{4.24}$$

La loi de probabilité est simple à déterminer dans le cas particulier des attributs qualité que nous avons choisis. En effet, vu que nous avons qu'une seule variable aléatoire, la loi de probabilité du degré de satisfaction est déduite directement à partir de la loi de probabilité du temps de réponse :

$$\forall z_k, z_k = F_\mu\left(u_1(x_i), u_2(q_{rel}), u_3(q_{av})\right), \text{ on a } \quad P_{DS}(z_k) = P_{X_1}(x_i) \ . \tag{4.25}$$

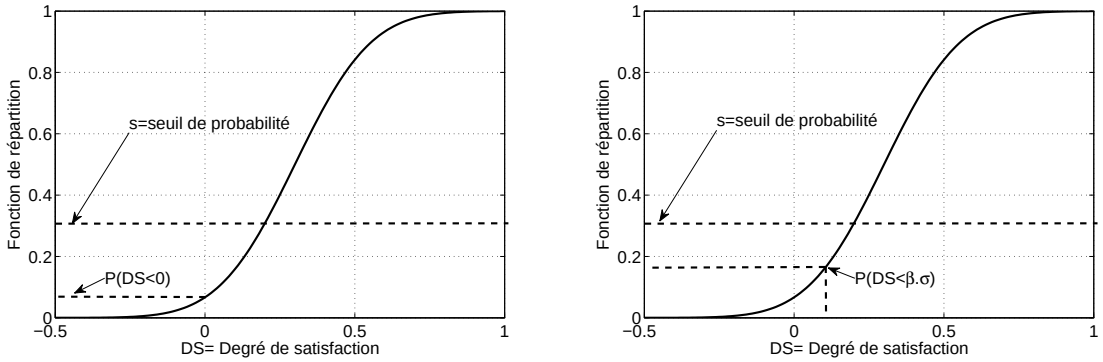
Ainsi, nous pouvons évaluer le degré de satisfaction lors des différentes phases du cycle de vie de l'orchestration, en particulier durant son exécution. Nous rappelons que l'objectif est de préserver le respect des exigences du client en termes de valeurs d'attributs qualité de l'orchestration. La question qui apparaît est comment contrôler le degré de satisfaction de l'orchestration tout au long de l'exécution. Cette question se décline en trois sous questions :

1. quand le système de supervision doit-il signaler qu'il y a une violation ou un risque d'une éventuelle violation des exigences du client ? Autrement dit, quand le système de supervision déclenche-t-il l'adaptation dynamique afin de remédier à la violation ou à la dégradation survenue ?
2. Quels attributs qualité faut-il améliorer en priorité ?
3. Quels sont les actions à mener pour rétablir le degré de satisfaction ?

Dans la section 4.5.2, nous répondons à la première des trois questions, la réponse à la deuxième question est l'objectif de nos futurs travaux tandis que la réponse à la dernière question est le sujet de la section 4.5.3.

4.5.2 Stratégies de surveillance

Rappelons que les exigences du client en termes de valeurs d'attributs qualité sont définies via les deux niveaux de qualité de référence $QoSL^{good}(orch)$ et $QoSL^{neutral}(orch)$, représentant respectivement le niveaux de qualité désiré et le niveau de qualité minimum acceptable. Ces deux niveaux de qualité sont respectivement associés aux degrés de satisfaction 1 et 0. Ceci implique que lorsque le degré de satisfaction appartient à l'intervalle [0,1], les exigences du client sont respectées. Par contre, un degré de satisfaction négatif signifie que les exigences du client sont violées. Dans ce contexte, nous proposons deux stratégies de surveillance : *une stratégie de surveillance réactive* et *une stratégie de surveillance préventive.*



(a) Cas de la surveillance réactive (b) Cas de la surveillance préventive

Figure 4.5 – Conditions de déclenchement de l'adaptation dynamique

Stratégie de surveillance réactive : elle consiste à déclencher l'évolution dynamique lorsque le degré de satisfaction (DS) viole les exigences du client en termes de valeurs d'attributs qualité, c'est-à-dire lorsqu'il devient inférieur à zéro ($DS < 0$). En réalité, nous n'avons pas qu'une seule valeur du degré de satisfaction durant l'exécution mais une variable aléatoire discrète définie par son domaine et sa loi de probabilité. Plus rigoureusement, la condition du déclenchement de l'évolution dynamique est dans ce cas :

$$P_{DS}(DS < 0) > s = \left(\sum_{\substack{z_k \in Dom(DS) \\ z_k < 0}} P_{DS}(z_k) \right) > s \tag{4.26}$$

où s est un seuil de probabilité (voir figure 4.5a). Vu que nous sommes dans un contexte probabiliste, ce seuil de probabilité peut être déduit dans la mesure où les contraintes globales (exprimées par le client) sur chaque attribut qualité de l'orchestration sont des contraintes probabilistes (voir ci-après). Dans le cas où le client établit des contrats (SLA) statistiques (voir section 2.4.1) avec les fournisseurs des services, nous avons aussi des contraintes probabilistes par attribut qualité et par service impliqué dans l'orchestration. En les agrégeant par les règles de patrons de *workflow*, nous pouvons obtenir des contraintes probabilistes sur chaque attribut qualité de l'orchestration. Au final, d'une manière ou d'une autre, la forme générale des contraintes sur chaque attribut qualité $i \in P$ de l'orchestration est la suivante :

$$P(a_i \leq X_i \leq b_i) = y_i \ .$$

Dans les contrats statistiques, nous pouvons avoir, dans certains cas, soit une minoration (a_i) lorsque l'attribut qualité i possède une dimension ascendante (voir section 2.3.1), soit une majoration (b_i) dans le cas contraire. Autrement dit, nous avons la probabilité que la valeur de l'attribut qualité i soit meilleure ou égale à la valeur minimum acceptable $q_i^{neutral}(orch)$. Supposons que nous avons m attributs qualité de dimension ascendante et

$(p-m)$ attributs qualité de dimension descendante. Nous avons donc :

- $P(X_i \geq q_i^{neutral}(orch))$, $i \in \{1,\ldots,m\}$, pour les attributs qualité de dimension ascendante ;
- $P(X_i \leq q_i^{neutral}(orch))$, $i \in \{m+1,\ldots,p\}$, pour les attributs qualité de dimension descendante.

Par conséquent, comme $DS = 0$ correspond à $F_\mu(u_1(q_1^{neutral}),\ldots,u_p(q_p^{neutral}))$, le seuil de probabilité s peut être déterminé par l'expression suivante :

$$s = 1 - P_{X1\ldots X_p}\Big(X_1 \geq q_1^{neutral}(orch),\ldots,X_m \geq q_m^{neutral}(orch), X_{m+1} \leq q_{m+1}^{neutral}(orch), \ldots, X_p \leq q_p^{neutral}(orch)\Big) \quad . \quad (4.27)$$

Sachant que les variables aléatoires $X_1,\ldots,X_p$ sont finies (les variables aléatoires continues sont supposées avoir été discrétisées, voir section 3.4.1), nous pouvons déterminer l'ensemble E :

$$E = \Big\{(x_1,\ldots,x_p) | \forall\, i \in \{1,\ldots,m\},\ x_i \geq q_1^{neutral}(orch), \forall i \in \{m+1,\ldots,p\}\ x_i \leq q_1^{neutral}(orch)\Big\} \quad .$$

En calculant la probabilité $P_{X1\ldots X_p}(x_1,\ldots,x_p)$ pour chaque $(x_1,\ldots,x_p) \in E$ à l'aide de l'expression 4.21, l'expression 4.27 devient :

$$s = 1 - \sum_{(x_1,\ldots,x_p)\in E} P_{X1\ldots X_p}(x_1,\ldots,x_p) \quad .$$

En l'absence de ces informations (c'est-à-dire contraintes probabilistes), le seuil de probabilité s peut être fixé par l'administrateur du système de supervision.

Exemple : prenons le cas du temps de réponse (X_1), de la fiabilité (X_2) et de la disponibilité (X_3). Supposons que les contraintes globales [7] (se rapportant à l'orchestration de services) spécifiées dans le contrat (SLA) sur ces trois attributs qualité sont les suivantes :

$$P(X_1 \leq q_{rt}^{neutral}) = y_1$$
$$X_2 \geq q_{rel}^{neutral}$$
$$X_3 \geq q_{av}^{neutral} \quad .$$

Dans ce cas, le seuil de probabilité est $s = 1 - y_1$.

Par ailleurs, il est possible de se passer du seuil de probabilité s en considérant la

7. Notons que la fiabilité et la disponibilité sont des valeurs fixes qui représentent des probabilités, il est donc incohérent de spécifier une probabilité d'avoir une valeur quelconque de fiabilité ou de disponibilité.

condition 4.26 de déclenchement suivante :

$$E[DS] < 0 \tag{4.28}$$

où $E[DS]$ est l'espérance mathématique de la variable aléatoire DS qui représente la valeur du degré de satisfaction attendue à long terme (la moyenne), elle est calculée de la manière suivante :

$$E[DS] = \sum_{z_k \in Dom(DS)} z_k.P_{DS}(z_k) \ .$$

Notons que la l'expression (4.28) n'implique pas l'expression (4.26) et vice versa, c'est-à-dire que l'on peut avoir l'expression (4.26) vérifiée sans que l'expression (4.28) le soit, etc. La condition (4.26) est plus rigoureuse, c'est pour cela que nous l'adoptons dans la suite comme condition de déclenchement de la surveillance réactive.

L'avantage de cette stratégie de surveillance réactive est qu'elle implique moins d'interventions sur l'instance de l'orchestration par rapport à la stratégie de surveillance préventive (voir ci-après). Par conséquent, elle est moins coûteuse. Par contre, son potentiel inconvénient est qu'elle peut être non efficace, notamment à un stade avancé de l'exécution de l'orchestration. En effet, d'une part, plus on se rapproche de la fin de l'exécution de l'orchestration, plus limitées sont les actions d'adaptation correctives. Par exemple, imaginons qu'au moment de déclenchement de l'adaptation il reste deux services à exécuter dans l'orchestration. Il est possible de substituer un ou les deux services, selon le besoin, avec un ou deux services équivalents de meilleurs niveaux de qualité. Par contre, s'il ne reste qu'un seul service non exécuté, nous n'avons le choix que d'agir sur ce dernier en le remplaçant par exemple par un autre service équivalent. D'autre part, il se peut qu'à un stade avancé de l'exécution, il ne soit plus possible de rétablir le degré de satisfaction du fait que la violation est trop importante et que la marge d'amélioration, apportée par les actions correctives, n'est pas suffisante. Ces limitations peuvent être surmontées en adoptant une stratégie de surveillance préventive.

Stratégie de surveillance préventive : cette stratégie consiste à spécifier un seuil d_0 de prévention pour assurer une marge de manœuvre. Le seuil d_0 peut être fixé par le client lors de la phase de conception. En effet, bien que le client ait choisi les services de telle sorte que le niveau de qualité perçu de l'orchestration respecte ses exigences, il peut ne pas être satisfait de son investissement lorsque le niveau de qualité perçu est plutôt proche du niveau de qualité minimum acceptable (limite inférieure de ses exigences). Nous offrons donc la possibilité au client de fixer ce seuil. En revanche, ce paramètre reste optionnel. Dans l'hypothèse où ce seuil n'est pas précisé par le client, nous procédons autrement. Nous nous sommes inspirés de la théorie de la maîtrise statistique des procédés (*Statistical Process Control-SPC*) où le procédé est considéré sous contrôle lorsque les mesures prélevées sont comprises dans la plage $[\mu - 3\sigma, \mu + 3\sigma]$, où μ et σ représentent ici respectivement la moyenne et l'écart-type de l'ensemble des mesures effectuées sur

le procédé. Si nous adaptons cette règle à notre contexte, l'adaptation dynamique est déclenchée lorsque, par exemple, la moyenne du degré de satisfaction, calculée au cours de l'exécution, devient inférieur à $\mu - 3\sigma$, où μ et σ sont respectivement la moyenne et l'écart-type « théoriques » du degré de satisfaction obtenus pendant la phase de conception. Or dans ce cas, cette borne inférieure $(\mu - 3\sigma)$ risque d'être elle même inférieure à zéro ce qui n'est pas cohérent pour une stratégie de surveillance préventive, d'où le rejet de ce seuil.

Au lieu de considérer la borne « 0 » pour déclencher l'adaptation comme c'est le cas de la surveillance réactive, nous proposons d'ajouter une marge de prévention proportionnelle à l'écart-type du degré de satisfaction. Ceci est traduit par la condition suivante (voir figure 4.5b) :

$$P_{DS}(DS < \beta.\sigma) > s \Leftrightarrow \left(\sum_{\substack{z_k \in Dom(DS) \\ z_k < \beta.\sigma}} P_{DS}(z_k) \right) > s \tag{4.29}$$

où β est un réel positif et σ est l'écart type de la variable aléatoire DS calculé comme suit :

$$\sigma = \sqrt{E[(DS - E[DS])^2]} = \sqrt{E[DS^2] - E[DS]^2}$$

Le paramètre β est un paramètre d'ajustement qui est fixé au préalable selon la criticité du processus métier ou l'exigence du client. Des grandes valeurs de β ($\beta \gg 1$) augmentent la fréquence de déclenchement de l'adaptation dynamique alors que les petites valeurs ($\beta < 1$) nous rapprochent du cas de la surveillance réactive.

Une fois que l'adaptation dynamique est déclenchée, il convient de définir des scénarios d'adaptation permettant d'améliorer la qualité de l'orchestration et rétablir le degré de satisfaction.

4.5.3 Adaptation dynamique des orchestrations

L'objectif de l'adaptation dynamique de l'orchestration au cours de son exécution consiste à effectuer les actions de correction nécessaires afin d'améliorer sa qualité globale, son degré de satisfaction dans notre cas. Il existe plusieurs actions, que nous appelons des scénarios d'évolution, permettant d'améliorer la qualité de l'orchestration. Parmi les scénarios d'évolution, nous pouvons citer la substitution, la modification du modèle de l'orchestration et la mise en place de stratégies de tolérance aux fautes.

La substitution : elle consiste à remplacer un ou plusieurs services par un ou plusieurs services équivalents au sens fonctionnel mais possédant une meilleure qualité de service. Ceci revient à un problème de sélection dynamique [Canfora et al., 2005b, Canfora et al., 2005a, Canfora et al., 2008] des services de la partie non exécutée de l'orchestration. Dans ce problème de sélection dynamique, les contraintes locales sur la qualité des services restent potentiellement les mêmes que celles spécifiées lors de la sélection pendant la phase de conception. Par contre, les contraintes globales sur la qualité de l'orchestration se résument en une seule contrainte selon la stratégie de surveillance adoptée :

- $P_{DS'}(DS' > 0) > s$ dans le cas de la surveillance réactive ;

(a) Ré-invocation (b) Séquence de services en attente

Figure 4.6 – Stratégies de tolérance aux fautes séquentielles

- $P_{DS'}(DS' > \beta.\sigma) > s$ dans le cas de la surveillance préventive.

où DS' représente la nouvelle variable aléatoire du degré de satisfaction qui est fonction des nouvelles variables aléatoires X'_i ($i \in P$) de l'orchestration globale. Chaque variable aléatoire X'_i est déterminée en agrégeant la mesure V_i relative à l'attribut qualité i de la partie exécutée de l'orchestration et les distributions se rapportant au même attribut qualité des (nouveaux) services de la partie non exécutée de l'orchestration (voir section 3.5). Le problème de la sélection des services est en dehors du périmètre de cette thèse.

La modification de la structure logique de l'orchestration : elle consiste à changer le chemin d'exécution de l'orchestration [Karastoyanova and Buchmann, 2004, Qiao et al., 2009] de manière à améliorer la qualité globale de l'orchestration. Ce scénario d'évolution nécessite la définition de tous les chemins d'exécution alternatifs et ce, pour tous les instants d'évolution possibles au cours de l'exécution comme c'est le cas dans [Qiao et al., 2009]. Nous pouvons aussi penser à réaliser un nouvel ordonnancement des fonctionnalités métier impliquées dans le processus de manière à respecter les contraintes, notamment lorsqu'il s'agit de contraintes de durée. Ce phénomène est connu par exemple, dans les problèmes d'ordonnancement en temps réel des systèmes manufacturiers pour gérer les aléas/pannes machines [Nasri et al., 2011]. Cependant, ces scénarios n'impactent potentiellement que les attributs qualité liés au temps. Les attributs qualité comme la fiabilité et la disponibilité, ne subissent pas d'impact puisque les opérations d'agrégation correspondantes impliquent souvent de la multiplication.

La mise en place des stratégies de tolérance aux fautes : ce scénario d'évolution permet d'améliorer particulièrement les attributs qualité de fiabilité et/ou de disponibilité. Il peut être choisi lorsque les poids d'importance (dans le modèle de préférence) relatifs à ces attributs qualité sont plus élevés par rapport aux autres (par exemple plus important que le poids d'importance du temps de réponse). [Zheng and Lyu, 2010] distinguent deux types de stratégies de tolérances aux fautes pour améliorer la fiabilité des services : stratégies séquentielles et stratégies parallèles. Il existe deux stratégies séquentielles [Zheng and Lyu, 2010] qui peuvent être mises en place.

La première stratégie séquentielle consiste à invoquer de nouveau la même opération de service pour un certain nombre de tentatives n (voir figure 4.6a). Cette stratégie ressemble au patron de *workflow* de boucle sauf qu'ici les tentatives d'invocation ne sont pas réalisées dans un but métier (répétition de la fonctionnalité n fois) mais elles sont dues à l'échec de

l'opération de service (non fiabilité). Les règles d'agrégation du temps de réponse et de la fiabilité pour cette stratégie sont [Zheng and Lyu, 2010] :

$$q_{rel} = 1 - (1 - q_{rel}(s))^n \qquad q_{rt} = \sum_{i=1}^{n} q_{rt_i}(s)(1 - q_{rel}(s))^{i-1} \tag{4.30}$$

où $q_{rt_i}(s)$ représente le temps de réponse de la $i^{ème}$ invocation de l'opération du service s. Dans ces règles, les tentatives d'invocation sont supposées indépendantes, c'est pour cela que les fiabilités des invocations sont toutes égales. [Sato and Trivedi, 2007] considèrent une fiabilité différente pour chacune des invocations. Dans ce cas, nous obtenons les règles d'agrégation suivantes :

$$q_{rel} = 1 - \prod_{i=1}^{n}(1 - q_{rel_i}(s)) \qquad q_{rt} = \sum_{i=1}^{n} q_{rt_i}(s) \prod_{k=1}^{i-1}(1 - q_{rel_k}(s)) \tag{4.31}$$

où $q_{rel_i}(s)$ représente la fiabilité de la $i^{ème}$ invocation de l'opération du service s. Cette stratégie étant appliquée pour la fiabilité, les invocations sont effectuées lorsque la réponse de l'opération de service est non fiable (voir section 2.3.1), la disponibilité globale reste neutre par rapport à cette stratégie. Autrement dit, la disponibilité de cette stratégie est déterminée de la même manière que pour le patron de *workflow* de boucle, soit $q_{av} = q_{av}(s)^n$. Cette stratégie a ses avantages et ses inconvénients. L'un de ses avantages est qu'elle n'est pas coûteuse par rapport à d'autres stratégies que nous présentons ci-après. Son inconvénient réside dans l'augmentation du temps de réponse comme le montre la règle d'agrégation. Cette stratégie d'invocation en boucle peut être également appliquée pour améliorer la disponibilité des opérations de service. Dans ce cas, nous proposons les règles d'agrégation suivantes pour estimer la disponibilité et le temps de réponse :

$$q_{av} = 1 - (1 - q_{av}(s))^n \tag{4.32}$$

$$q_{rt} = \sum_{i=1}^{n} q_{av}(s)(1 - q_{av}(s))^{i-1}\Big((i-1)t_0 + q_{rt}(s)\Big) + (1 - q_{av}(s))^n(n \cdot t_0) \tag{4.33}$$

où t_0 représente le dépassement de délai (*time out*) dû à l'indisponibilité de l'opération de service (un délai entre deux tentatives d'invocation peut être également envisagé et inclus dans t_0). La règle relative au temps de réponse est la somme de deux termes : le premier terme correspond au temps de réponse lorsque nous avons $(i-1)$ indisponibilités suivies d'une disponibilité à la $i^{ème}$ invocation et ceci pour tout i compris entre 1 et n. L'occurrence de cet évènement est traduite par la probabilité $q_{av}(s)(1 - q_{av}(s))^{i-1}$. Le temps de réponse correspondant est par conséquent la somme de $(i-1)t_0$ (puisque nous avons $(i-1)$ indisponibilités) et du temps de réponse $q_{rt}(s)$ de l'opération de service lorsqu'elle est disponible. Le deuxième terme dans la règle d'agrégation correspond au cas où il y a eu n indisponibilités. Dans ce cas, le temps consommé par les n invocations est $(n \cdot t_0)$. Là encore, nous avons supposé que les invocations sont indépendantes. Dans le cas

contraire, les règles d'agrégation changent légèrement :

$$q_{av} = 1 - \prod_{i=1}^{n}(1 - q_{av_i}(s)) \tag{4.34}$$

$$q_{rt} = \sum_{i=1}^{n}\left(q_{av_i}(s)\Big(\prod_{k=1}^{i-1}(1 - q_{av_k}(s))\Big)\Big((i-1)t_0 + q_{rt}(s)\Big)\right) + \prod_{i=1}^{n}(1 - q_{av_i}(s)) \cdot n \cdot t_0 \tag{4.35}$$

où $q_{av_i}(s)$ représente la disponibilité de la $i^{ème}$ invocation. D'autre part, si les invocations sont réalisées seulement dans le cas de l'indisponibilité de l'opération de service, la fiabilité globale de cette stratégie est déterminée comme dans le cas du patron de *workflow* boucle, soit $q_{rel} = q_{rel}(s)^n$. Nous proposons finalement de combiner les deux cas ; les invocations sont effectuées soit dans le cas où l'opération de service est indisponible, soit lorsque la réponse retournée n'est pas fiable. Ainsi, nous définissons les règles d'agrégation correspondantes, pour les trois attributs qualité :

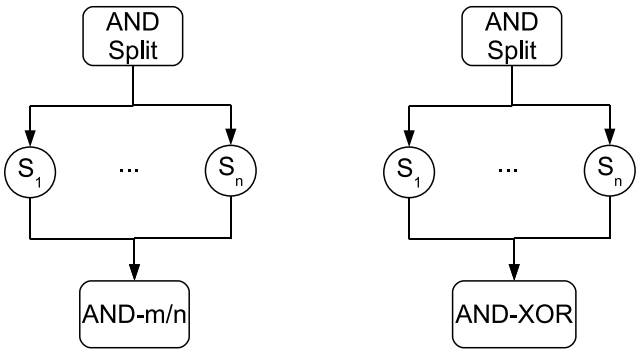
$$q_{rel} = 1 - (1 - q_{rel}(s))^n \tag{4.36}$$

$$q_{av} = 1 - (1 - q_{av}(s))^n \tag{4.37}$$

$$q_{rt} = q_{av}(s)q_{rel}(s)\sum_{i=1}^{n}\sum_{k=0}^{i-1}(1 - q_{av}(s))^{i-k-1}(1 - q_{rel}(s))^k\Big[(i-1-k)t_0 + (k+1)q_{rt}(s)\Big] + \sum_{k=0}^{n}(1 - q_{av}(s))^{n-k}(1 - q_{rel}(s))^k\Big[(n-k)t_0 + kq_{rt}(s)\Big] \tag{4.38}$$

Nous constatons que la règle d'agrégation du temps de réponse est composée d'une somme de deux termes. Le premier terme correspond au temps de réponse résultant d'un succès de l'invocation à la $i^{ème}$ tentative qui peut être obtenu en ayant k réponses non fiables et $(i-1-k)$ indisponibilités de l'opération de service, avec $0 \leq k \leq i-1$ et $1 \leq i \leq n$. Le second terme correspond au temps de réponse suite à un échec total au bout de n invocations qui résulte de k, $0 \leq k \leq n$, réponses non fiables et $(n-k)$ indisponibilités de l'opération de service. Cette stratégie permet donc d'améliorer à la fois la disponibilité et la fiabilité mais elle a un impact qui peut être non négligeable sur le temps de réponse.

La seconde stratégie séquentielle consiste à mettre en séquence, avec le service en question, d'autres services en attente (*standby*) qui sont équivalents au sens fonctionnel (voir figure 4.6b). L'opération du service i est invoquée lorsque la réponse de l'opération du service $(i-1)$ est non fiable. Les règles d'agrégation de cette stratégie sont identiques à celles de la stratégie précédente où les invocations ne sont pas indépendantes (voir équations 4.31). Le même raisonnement fait pour la stratégie précédente s'applique ici, c'est-à-dire utiliser cette stratégie pour améliorer aussi la disponibilité ou encore pour améliorer, à la fois, la fiabilité et la disponibilité. Les règles d'agrégation pour ces situations sont semblables à celles de la stratégie séquentielle d'invocation en boucle, c'est pour cette raison que nous les omettons. L'avantage de cette stratégie par rapport à la précédente est qu'elle



(a) Patron de composition AND-m/n

(b) Patron de composition AND-XOR

Figure 4.7 – Stratégies de tolérance aux fautes parallèles

est plus efficace. En effet, elle présente plus de flexibilité : les services en attente peuvent être choisis de telle sorte que leur disponibilité et/ou fiabilité sont meilleures que celle du premier service. En outre, il y a plus de chance que les services en attente accomplissent leur tâche (au sens de la fiabilité et de la disponibilité) contrairement à l'invocation en boucle où souvent, lorsque la $i^{ème}$ tentative échoue (pour des raisons d'indisponibilité ou de non fiabilité), il est plus probable que la tentative suivante échoue aussi. Par contre, le nombre de services en attente implique plus d'investissement et donc cette stratégie est plus coûteuse que la précédente. D'autre part, elle présente le même inconvénient que la stratégie séquentielle d'invocation en boucle pour ce qui est de l'augmentation du temps de réponse.

En ce qui concerne les stratégies parallèles de tolérance aux fautes, nous les avons déjà introduites dans le chapitre précédent. Ce sont les patrons de composition *AND-m/n* (CP6) et *AND-XOR* (CP7) où les branches en parallèle contiennent des services équivalents au sens fonctionnel (voir figure 4.7). Les règles d'agrégation relatives à ces patrons de compositions ont été déjà présentées dans le chapitre précédent (voir section 3.3) pour chacun des attributs qualité considérés. La différence entre ces deux patrons de composition est que le patron CP6 est plus exigent en terme de fiabilité ; il faut que m réponses soient identiques pour s'assurer de la fiabilité du résultat. Ces deux patrons de composition consomment moins de temps de réponse que les stratégies séquentielles, en particulier le patron CP7 est le plus rapide puisque son temps de réponse est le minimum de tous les temps de réponse des services en parallèle. Cependant, ces stratégies parallèles sont plus coûteuses que les stratégies séquentielles du fait que tous les services sont exécutés simultanément contrairement à la stratégie séquentielle, par exemple lorsque les services sont en séquence, dans laquelle le service i n'est exécuté que si le service $(i-1)$ est non fiable ou indisponible.

Ces règles d'agrégation que nous avons proposées, ciblent la phase de conception et donnent une estimation des attributs qualité considérés. Notons que pour le temps de ré-

ponse, la règle d'agrégation est appliquée pour chaque valeur x_i appartenant au domaine de la variable aléatoire discrète représentant le temps de réponse. Comme nous nous intéressons particulièrement à la phase d'exécution de l'orchestration, il convient d'évaluer les attributs qualité pour les stratégies de tolérances aux fautes présentées. Pour cela, nous allons proposer des métriques pour évaluer la fiabilité et la disponibilité de chaque stratégie en s'inspirant des métriques relatifs à ces attributs qualité (voir section 2.3.1), le temps de réponse étant mesuré directement à l'exécution par la fonction d'acquisition du système de supervision (voir figure 3.1) :

- stratégie séquentielle d'invocation en boucle :

$$q_{rel} = \frac{n_S}{n_T} \tag{4.39}$$

$$q_{av} = 1 - \frac{n_1 \times t_0}{n_1 \times t_0 + n_2 \times q_{rt}} \tag{4.40}$$

  où :
  - n_S est le nombre d'invocations de l'opération de service en question, réalisées avec succès dans la partie exécutée de l'orchestration ;
  - n_T est le nombre total d'invocations de l'opération de service considérée, dans la partie exécutée de l'orchestration ;
  - t_0 est le dépassement de délai (*time out*) moyen observé (rapporté par la fonction d'acquisition du système de supervision, voir figure 3.1) de l'opération de service pendant la partie exécutée de l'orchestration ;
  - n_1 est le nombre de fois que l'on a observé un dépassement de délai dans la partie exécutée de l'orchestration ;
  - n_2 est le nombre de fois que l'opération de service était disponible dans la partie exécutée de l'orchestration ;
  - q_{rt} est le temps de réponse moyen mesuré de l'opération de service dans la partie exécutée de l'orchestration.
- Stratégie séquentielle de services en attente :

$$q_{av} = 1 - \frac{\sum_{i=1}^{n} n_i \times t_0(S_i)}{\sum_{i=1}^{n} n_i \times t_0(S_i) + \sum_{j=1}^{m} \sum_{i=1}^{n} p_i . q_{rt}(S_i)} \tag{4.41}$$

  où :
  - $t_0(S_i)$ est le dépassement de délai de l'opération de service S_i ;
  - n_i est le nombre de fois que l'opération de service S_i a été indisponible depuis le début de l'exécution ;
  - $q_{rt}(S_i)$ est le temps de réponse moyen mesuré de l'opération de service S_i ;
  - p_i prend la valeur de 1 si l'opération de service S_i a été exécutée et 0 dans le cas contraire ;
  - m est le nombre de fois que la stratégie de tolérance aux fautes, pour l'opération de service en question, a été exécuté depuis le début de l'exécution.

  L'évaluation de la fiabilité se fait de la même façon que dans le cas de la stratégie

d'invocations en boucle sauf que le sens de n_S et n_T changent légèrement. Dans ce cas, n_S désigne le nombre d'invocations effectuées avec succès incluant toutes les opérations de service S_i impliquées dans la stratégie et ce, depuis le début de l'exécution. Quant au paramètre n_T, il représente le nombre total d'invocations depuis le début de l'exécution, incluant toutes les opérations de service impliquées dans la stratégie.

- Stratégies parallèles :

$$q_{av} = 1 - \frac{\sum_{i=1}^{n} n_i \times t_0(S_i)}{\sum_{i=1}^{n} n_i \times t_0(S_i) + \sum_{j=1}^{m} q_{rt}(j)} \tag{4.42}$$

où :

– $t_0(S_i)$ est le dépassement de délai de l'opération de service S_i ;
– n_i est le nombre de fois que l'opération de service S_i a été indisponible depuis le début de l'exécution ;
– $q_{rt}(j)$ est le temps de réponse de la stratégie parallèle en question pendant son $j^{ème}$ exécution ;
– m est le nombre de fois que la stratégie de tolérance aux fautes a été exécutée depuis le début de l'exécution.

L'évaluation de la fiabilité est identique à celle de la stratégie séquentielle de services en séquence.

4.6 Synthèse

Nous avons présenté dans ce chapitre une approche pour mesurer le degré de satisfaction des orchestrations de services au regard des exigences des clients en terme de qualité. Pour ce faire, nous avons appliqué dans un premier temps la méthode MACBETH (section 4.3). Nous avons montré que cette méthode permet de construire les fonctions d'utilité marginales et la fonction d'utilité globale (c'est-à-dire déterminer les paramètres de l'opérateur d'agrégation) à partir des informations (préférences et intensités de préférences) fournies par le client. De cette manière le modèle de préférences est plus précis au sens où il représente mieux la satisfaction réelle des clients. Cependant, nous avons vu que la méthode MACBETH n'autorise pas la prise en compte des dépendances, en particulier les dépendances préférentielles. L'exemple présenté dans la section 4.4.1 illustre cette problématique. Pour remédier à cette limitation et afin de construire un modèle de préférences traduisant au mieux les préférences du client, nous avons appliqué la méthode MACBETH étendue avec l'opérateur de l'intégrale de Choquet 2-additive (section 4.4.3). Compte tenu de ces deux facteurs, la prise en compte des préférences du client et des dépendances préférentielles, nous estimons que notre approche fournit de meilleurs résultats par rapport aux travaux existants dans la littérature. Par exemple, [Canfora et al., 2008, Taher et al., 2005b] utilisent l'opérateur d'agrégation de la moyenne pondérée et des équations linéaires pour l'agrégation des attributs qualité et leur normalisation. Bien que [Herssens

et al., 2008] aient exploité l'opérateur de l'intégrale de Choquet 2-additive, leur fonctions d'utilité marginales sont des équations linéaires. Nous allons effectuer dans le chapitre suivant (voir section 5.4.1) une comparaison de notre modèle de préférences par rapport à ces différents travaux de la littérature.

Ainsi, l'approche globale d'agrégation que nous avons proposée, l'agrégation par les règles de patrons de *workflow* (voir chapitre 3) combinée à l'agrégation par une méthode d'aide à la décision multi-critères, permet au système de supervision de mesurer le ressenti du client par rapport à la qualité perçue de l'orchestration. De surcroît, la présence d'une seule information de haut niveau, *le degré de satisfaction*, permet au système de supervision de faciliter non seulement l'interprétation des valeurs des attributs qualité, mais aussi la prise de décision quant aux détections des anomalies/dégradations de la qualité de l'orchestration, notamment au cours de l'exécution. Dans ce sens, nous avons proposé deux stratégies de surveillance (réactive et préventive) et les conditions respectives de déclenchement de l'adaptation dynamique (voir section 4.5.2). Ces conditions, quelle que soit la stratégie de surveillance adoptée, sont basées sur le degré de satisfaction de l'orchestration. Nous avons ensuite étudié différents scénarios d'évolution qui sont susceptibles d'améliorer la qualité de l'orchestration. En particulier, nous avons détaillé plusieurs stratégies de tolérance aux fautes permettant d'améliorer la fiabilité (voir section 4.5.3). Nous avons étendu ces stratégies pour la disponibilité et nous avons proposé des règles d'agrégation permettant de prendre en compte simultanément la fiabilité et la disponibilité.

Chapitre 5

Application de l'approche de supervision à une architecture orientée service d'une solution de pilotage d'ateliers de production

5.1 Introduction

Nous avons présenté jusqu'ici une approche de supervision des orchestrations de services. L'intérêt majeur de cette approche est la supervision des orchestrations de services tout au long de leur exécution. Étant donné une orchestration de services, l'approche consiste à évaluer *le degré de satisfaction* de l'orchestration tout au long de son exécution et à gérer dynamiquement son adaptation lorsque les exigences du client (en termes de qualité) ne sont plus respectées. Lors de la présentation de cette approche, notre point de départ était, entre autres, l'existence d'un modèle d'orchestration de services. En particulier, nous avons toujours supposé que les services et leurs opérations de service sont bien définis et sont exposés par des fournisseurs de services (entreprises, organisations, etc). Dans ce chapitre, nous nous focalisons dans un premier temps sur la définition des services et leurs opérations de service dans un domaine particulier qui est le pilotage d'aterliers de production (en anglais *Manufacturing Execution System, MES*). Plus globalement, nous présentons la mise en œuvre d'une architecture orientée service pour une solution logicielle dédiée au MES. L'application de nos travaux a été réalisée dans le cadre d'un projet de recherche et développement régional regroupant plusieurs entreprises (section 5.2). Ensuite, nous considérons un exemple de scénario MES (section 5.3) sur lequel, nous appliquons notre approche de supervision pendant la phase de conception (section 5.4) et au cours de l'exécution du scénario (section 5.5).

5.2 Application des architectures orientées services dans le cadre du projet MES

5.2.1 Contexte du projet MES

Une partie de nos travaux a été réalisée dans le cadre d'un projet de recherche de la région Rhône-Alpes. Le projet MES (*Manufacturing Execution System*), est un projet de Recherche et Développement [Khamès, 2010] financé en partie par la région Rhône-Alpes et soutenu par les clusters Edit et GOSPI [1]. Ce projet collaboratif fédère sept éditeurs de progiciels dans le domaine du MES et deux laboratoires de recherche. Ces différents éditeurs couvrent des besoins métiers variés, majoritairement différents et parfois concurrents dans le domaine du MES. Dans ce contexte, les principaux objectifs ciblés par le projet consistent à concevoir et développer une solution logicielle répondant au marché du MES, qui soit :

- complète, (couvrant tout le domaine du MES) en exploitant les domaines de compétence et les offres métiers des différents éditeurs de progiciels impliqués dans le projet ;
- agile, permettant au consortium d'entreprises de répondre à des besoins spécifiques/variés des clients et de couvrir un marché plus large. L'agilité permet également aux clients de se doter d'une flexibilité et d'une souplesse dans l'adaptation de leurs processus métiers. On parle parfois « des processus à la carte » pour faire référence à l'agilité de cette solution logicielle ;
- interopérable, en respectant au mieux les standards technologiques et les normes du domaine du MES.

Dans ce contexte et afin de répondre en grande partie à ces objectifs, nous avons opté pour une architecture orientée services (voir section 2.2) comme architecture de la solution logicielle MES [Verjus et al., 2011].

5.2.2 Architecture de la solution MES

Rappelons que les architectures orientées services se basent sur trois entités : le client, le fournisseur de service et l'annuaire de services (voir figure 2.1). Dans le contexte du MES, le client symbolise le(s) processus métier(s) du MES que l'entreprise (cliente) veut décrire à base de services. L'annuaire de services n'est qu'une exposition des services et des fonctionnalités qu'offrent les éditeurs de progiciels. Autrement dit, l'annuaire de services représente une cartographie des services et leurs opérations de service assurant différents besoins/fonctionnalités métiers et qui sont cartographiés selon des domaines métiers particuliers (voir section 5.2.3). Quant aux fournisseurs des services, ils représentent les entreprises partenaires dans le projet qui implémentent les fonctionnalités présentes dans la cartographie de services sous forme de services et opérations de service (voir figure 5.1). Le premier travail engagé nous a permis de décliner cette architecture en trois niveaux

1. http://www.industrie.com/it/conception/projet-mes-une-gestion-d-atelier-a-la-carte-pour-pme.7937

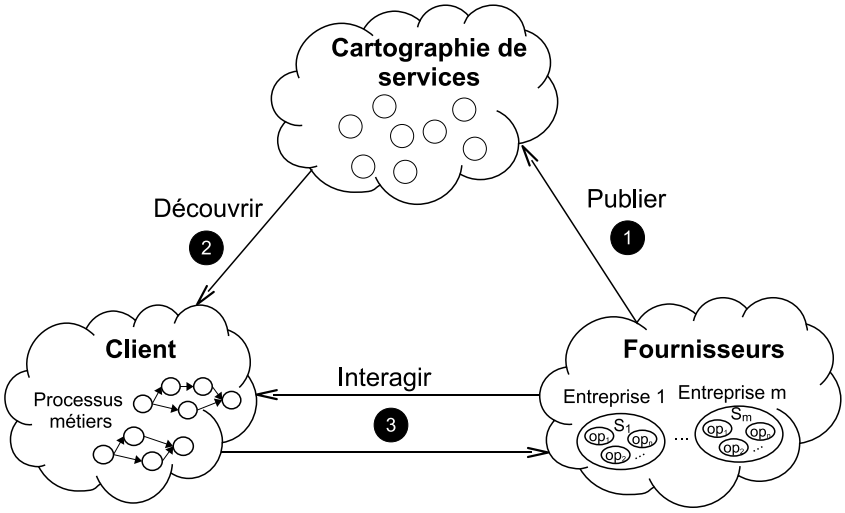


Figure 5.1 – Principe des AOS appliqué dans le cadre du projet MES

selon une approche d'Ingénierie Dirigée par les Modèles (IDM) [Verjus et al., 2011] (voir figure 5.2) :

- *le niveau métier* comprend une ou un ensemble de représentations de l'entreprise cliente (du MES) et en particulier, l'expression des processus de gestion de production/pilotage d'ateliers ; ces représentations sont exprimées par des modèles d'entreprise. Le niveau métier correspond au niveau *Computer Independent Model* (CIM) de l'approche IDM ;
- *le niveau fonctionnel* regroupe l'ensemble des fonctionnalités du domaine du MES exposées sous forme de services (logiciels) et opérations de service. Ce niveau fonctionnel peut aussi comprendre des orchestrations de services abstraites qui constituent dans ce cas, une projection des processus du niveau métier. On entend par orchestration abstraite, l'orchestration des services fonctionnels (ou orchestration de fonctionnalités) indépendamment de leur implémentation. Le niveau fonctionnel correspond au niveau *Platform Independent Model* (PIM) de l'approche IDM ;
- *le niveau technologique* regroupe l'ensemble des services et des opérations de services MES, proposés par les éditeurs de progiciels impliqués dans le projet. Dans le cadre du projet, les services et leurs opérations de services sont exposés sous forme de services Web respectant les standards du Web 2.0. Les opérations de service de ce niveau technologique sont orchestrées conformément aux modèles d'orchestrations définis dans le niveau fonctionnel qui, eux-mêmes, supportent les processus du niveau métier. Notons que ce sont les services et les opérations de service de ce niveau technologique qui sont exécutés au final par le moteur d'exécution des orchestrations. Le niveau technologique représente le niveau *Platform Specific Model* (PSM) de l'approche IDM.

La définition des modèles du niveau métier (CIM) était prise en charge par un des partenaires du projet MES. D'autre part, le passage d'un niveau à un autre n'a pas été automatisée, elle ne rentre pas dans le périmètre du projet. L'automatisation de ces pas-

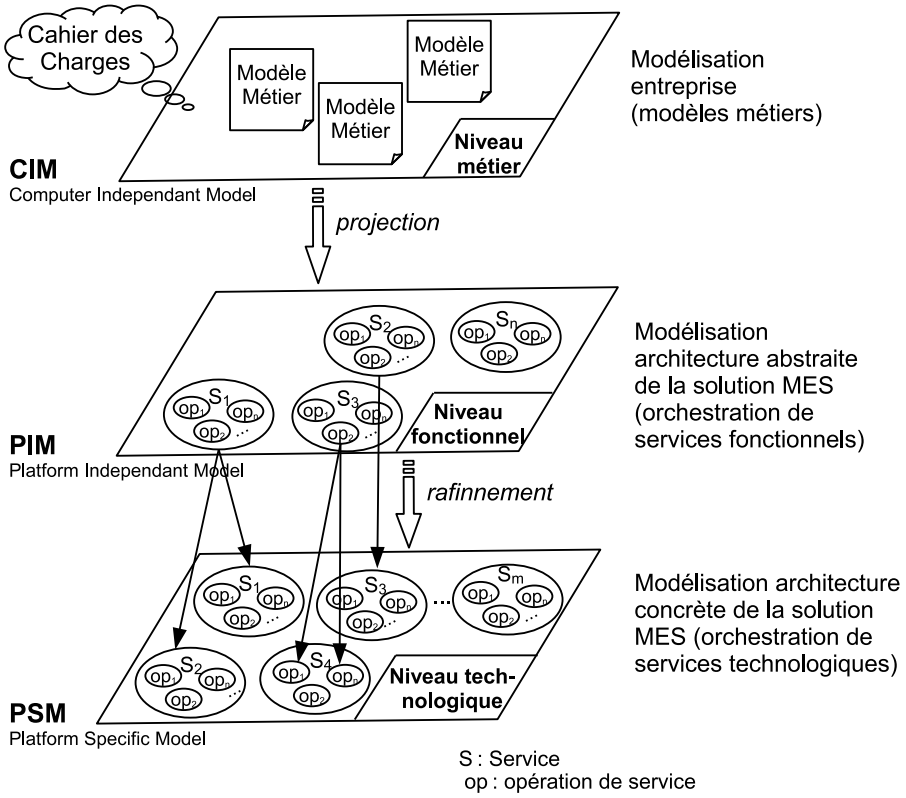


Figure 5.2 – Approche IDM pour l'architecture logicielle de la solution MES

sages (transformations de modèles) est l'objectif des futurs travaux au sein du projet. Dans la suite, nous nous focalisons essentiellement sur les niveaux fonctionnel et technologique. Pour cela, nous avons identifié les tâches suivantes à mener :

- établir une cartographie des services fonctionnels du MES : toutes les fonctionnalités offertes par les différents partenaires dans le domaine du MES doivent être recensées et exposées sous forme de services et leurs opérations de service ;
- identifier les paramètres d'entrée/sortie pour chacune des fonctionnalités (opérations de service) et les représenter conformément aux normes et référentiels du MES ;

5.2.3 Cartographie des services MES

La cartographie MES vise principalement deux niveaux que nous avons appelés niveau fonctionnel et niveau technologique.

Niveau fonctionnel de la cartographie MES

Nous nous sommes engagés dans le cadre du projet MES, à définir une architecture orientée services pour la solution logicielle MES. Pour atteindre cet objectif, les fonctionnalités MES de l'ensemble des progiciels doivent être exposées sous forme de services qui pourront être invoqués et orchestrés à la demande. Pour cela, nous avons commencé dans un premier temps, par explorer le domaine du MES et recenser les différentes fonction-

nalités que fournissent les progiciels des éditeurs dans le domaine du MES. Ce travail s'est déroulé principalement sous forme d'interviews et d'ateliers que nous avons menés auprès de chacun des éditeurs participant au projet MES. L'exploitation des informations que nous avons recueillies lors de nos rencontres avec les éditeurs, nous a permis de dégager une première liste de fonctionnalités MES. Ces fonctionnalités doivent respecter les propriétés des architectures orientées services dont *la propriété de faible couplage*, qui est particulièrement importante. L'objectif communément admis du faible couplage, est d'introduire le minimum de dépendances entre des services différents pour permettre de les assembler (orchestrer) aisément [Collet, 2006, Pourraz, 2007] à la demande et de façon indépendante. Ceci implique que des opérations d'un même service peuvent avoir des dépendances (fonctionnelles) entre elles alors que deux opérations de service appartenant à deux services différents doivent être faiblement couplées (c'est-à-dire avoir le minimum de dépendances possibles). Partant de ce principe, la tâche suivante dans l'élaboration de la cartographie de services MES, a consisté à recenser les dépendances fonctionnelles entre les fonctionnalités MES. Sachant que les fonctionnalités MES vont être représentées par des opérations de service, l'objectif est donc de regrouper ces opérations en services qui soient faiblement couplés. Pour cela, nous avons dressé une matrice des dépendances entre les fonctionnalités MES que nous avons prises en compte dans le regroupement des opérations de services. Un autre critère que nous avons pris en compte dans le regroupement des opérations de service, est le sens ou le besoin métier : un service doit avoir un sens métier (pour qu'il soit significatif vis-à-vis du client MES) et répondre à un besoin métier spécifique. Par conséquent, les opérations de service appartenant à un même service doivent converger vers ce sens/besoin métier.

D'autre part, l'un des objectifs du projet est que la solution logicielle MES soit conforme aux normes du domaine MES, en particulier à la norme ISA-95 [B.V., 2012]. La norme ISA-95 est une norme américaine développée par l'ISA (*Internationale Society of Automation*). Elle est également publiée comme une norme internationale ISO/IEC62264 et une norme française NF EN 62264 [ISA-France, 2012]. La norme ISA-95 consiste en des modèles et terminologies qui sont utilisés pour déterminer et décrire les informations échangées entre les différents systèmes au sein du MES comme le système de production, le système de maintenance, le système de qualité, etc. Elle est également utilisée comme une base pour développer des interfaces entre le système MES et les progiciels de gestion intégrée (*Enterprise Ressource Planning–ERP*). La norme ISA-95 distingue quatre domaines du MES qui sont la production, la maintenance, la qualité et la gestion du stock (logistique). Pour chacun de ces domaines, la norme présente un modèle d'activité générique qui comporte huit fonctions :

1. l'ordonnancement détaillé ;
2. la gestion des ressources ;
3. la distribution des ressources (*dispatching*) ;
4. la traçabilité ;

5. la définition des produits ;
6. la gestion de l'exécution ;
7. la collecte et l'acquisition de données ;
8. l'analyse.

La première idée a été de classifier les fonctionnalités que nous avons recensées selon ces quatre domaines et ces huit fonctions et de les regrouper ensuite en services. Cependant, il s'est avéré cependant que cette classification n'est pas pertinente dans le sens où elle ne respecte pas les critères que nous avons fixés précédemment, à savoir le faible couplage entre les services et le sens métier que doit refléter un service. À titre d'exemple, « Créer un ordre de fabrication » et « Lancer un ordre de fabrication » sont deux fonctionnalités qui s'inscrivent respectivement dans « l'ordonnancement détaillé » et « la gestion de l'exécution » du domaine de la production. Or, ces deux fonctionnalités sont dépendantes techniquement du fait que le lancement d'un ordre de fabrication ne peut être effectué que s'il a été déjà créé auparavant dans le même système. Par conséquent, ces deux fonctionnalités (opérations de service) doivent appartenir au même service. En outre, nous avons identifié des dépendances au sens fonctionnel (métier) entre des fonctionnalités appartenant à des domaines, définis par la norme ISA-95, différents. Par exemple, la fonctionnalité « Créer un équipement » peut appartenir :

- au domaine de la production, lorsqu'il s'agit d'une machine de production ;
- au domaine de la qualité, lorsqu'il s'agit d'une machine liée au contrôle qualité ;
- au domaine de la maintenance, pour la gestion de la maintenance des équipements.

Ceci nous a conduit à opter pour une classification différente des fonctionnalités MES et privilégier le découpage des fonctionnalités selon leur sens métier tout en respectant la propriété de faible couplage. Ainsi, afin de surmonter les difficultés de découpage présentées précédemment, nous avons décidé de répartir les fonctionnalités MES selon les quatre domaines définis par la norme et de rajouter un cinquième domaine transversal qui englobe toutes les fonctionnalités transversales aux quatre domaines, comme par exemple la gestion documentaire ou la gestion des équipements. Ensuite, nous avons regroupé les fonctionnalités de chacun des domaines en services de telle manière qu'ils aient un sens métier, comme par exemple la gestion des ordres de fabrication, la planification de la production, etc. Ce travail a été réalisé en collaboration étroite avec les partenaires du projet MES. Après la validation des éditeurs de progiciels, la cartographie [2] des services MES comprend aujourd'hui 242 opérations de service regroupées en 23 services [Fakhfakh et al., 2011c].

Après avoir identifié cet ensemble d'opérations de service, nous nous sommes chargés d'identifier leurs paramètres d'entrée/sortie. En réalité, un sous ensemble d'opérations de service a été sélectionné par les éditeurs comme les opérations de service prioritaires. Cette étape a été réalisée en collaboration avec les éditeurs de progiciels sous forme d'ateliers de travail que nous avons organisés, et a permis d'identifier 1962 paramètres d'entrée/sortie

2. Le consortium doit se prononcer sur la mise à disposition (publique ou non) de la cartographie.

pour les opérations de service sélectionnées. La seconde étape a permis de structurer ces paramètres d'entrée/sortie tel que le définit la norme ISA-95. En effet, *Business To Manufacturing Markup Language* (B2MML) [WBF, 2012] est l'implémentation XML du standard ISA-95. B2MML consiste en un ensemble de schémas XML qui implémentent les modèles de la norme ISA-95. Il définit des structures de données typées et rigoureusement nommées qui permettent d'avoir une terminologie commune et compréhensible par les différents systèmes qui communiquent entre eux. L'utilisation des schémas B2MML dans le cadre du projet MES nous permet non seulement de favoriser l'interopérabilité entre les différents systèmes au sein du MES mais aussi de favoriser l'interopérabilité avec d'autres systèmes au sein de l'entreprise comme les ERP, etc. Pour cela, les paramètres d'entrée/sortie des opérations de service que nous avons identifiées, ont été décrits par le biais des schémas B2MML mettant en œuvre la norme ISA-95 [DISP, 2011]. De cette manière, le niveau fonctionnel de la cartographie des services MES est indépendant de toute implémentation particulière.

Niveau technologique de la cartographie MES

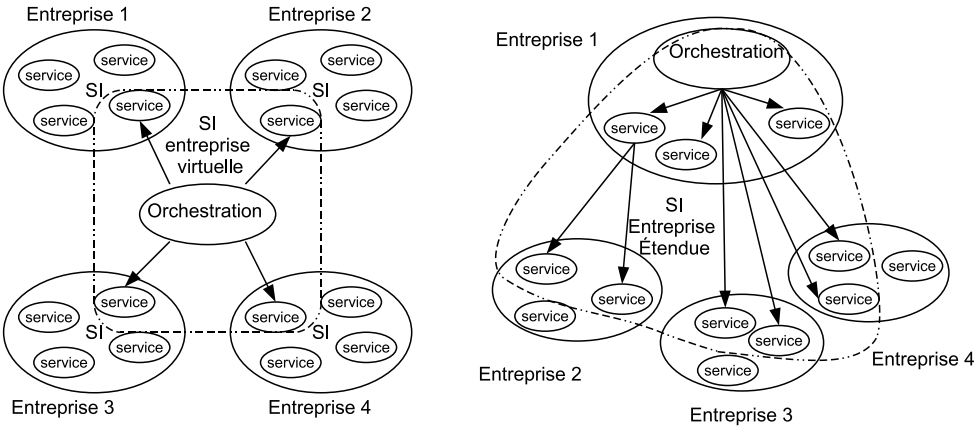
Les services du niveau fonctionnel définis dans la cartographie que nous avons élaborée, sont amenés à être supportés par les services du niveau technologique. Les services du niveau technologique sont ceux proposés par les éditeurs des progiciels. Ils peuvent supporter directement les services fonctionnels. Dans ce cas, ils représentent une implémentation particulière des services fonctionnels. Ils peuvent aussi être à un niveau de granularité plus fin auquel cas une opération de service du niveau fonctionnel est concrétisée par une orchestration de services du niveau technologique. Dans le cadre du projet MES, c'est souvent le premier cas qui se présente. Les services du niveau technologique sont donc une implémentation particulière des services du niveau fonctionnel, à savoir *des services Web*. Pour cela, il a été nécessaire de décrire les interfaces (WSDL) des services Web et en particulier les paramètres d'entrée/sortie de chaque opération de service [Fakhfakh et al., 2011a]. Dans ce sens, nous avons produit les interfaces (fichiers WSDL) des services avec les standards du Web 2.0. Ces interfaces décrivant les services, leurs opérations ainsi que les paramètres d'entrée/sortie de ces opérations de service, ont permis aux éditeurs de progiciels de développer les services Web qui sont fournis dans la solution logicielle MES, appelée MESTRIA [Verjus, 2011].

Par ailleurs, plusieurs scénarios (processus) métiers du domaine du MES ont été recensés pour répondre à des besoins variés, comme par exemple un processus de production à l'affaire, un processus de contrôle à la réception, un processus de maintenance, etc. La définition des orchestrations de services supportant ces processus métiers s'est faite en considérant dans un premier temps les services fonctionnels de la cartographie MES, puis les services Web proposés par les éditeurs. Ceci correspond à la partie grisée de la figure 3.1 (voir section 3.2.2, page 52) appliquée au domaine du MES, en particulier dans le cadre du projet MES. Le fruit de ces travaux a permis le développement d'un prototype de la

solution logicielle MES, appelé MESTRIA [Verjus, 2011]. Cette solution logicielle, appelée MESTRIA, est amenée à être mise en place chez des clients MES et par la suite à être intégrée dans le système d'information de l'entreprise cliente. Pour cela, nous avons identifié plusieurs cas de figure que nous présentons dans la section suivante.

5.2.4 Intégration de la solution logicielle MES dans un système d'information à base de services

L'architecture orientée services que nous avons mise en œuvre dans le cadre du projet MES, permet non seulement d'avoir une flexibilité et une agilité pour concevoir la solution MES mais aussi de favoriser une certaine ouverture vers l'extérieur, en particulier son intégration dans le système d'information d'une entreprise tierce. Ceci est d'autant plus vrai lorsque le système d'information est à base de services [Fakhfakh et al., 2010, Verjus et al., 2011]. La démarche présentée précédemment (voir section 5.2.3) peut être suivie pour appliquer les architectures orientées services aux systèmes d'information des entreprises. À titre d'exemple, la solution MES elle-même peut être vu comme un système d'information puisqu'elle combine plusieurs progiciels (hétérogènes) de différents éditeurs. Dans la section précédente, nous avons parlé de la collaboration de l'ensemble des éditeurs permettant de fournir une solution MES à base de services. Dans ce cas, l'ensemble des éditeurs mettent en commun leurs services qui seront orchestrés afin de réaliser des processus collaboratifs inter-entreprises. On parle alors d'entreprise virtuelle regroupant l'ensemble des éditeurs (voir figure 5.3a) [Fakhfakh et al., 2010]. Il est possible par ailleurs qu'un des éditeurs partenaires du projet ait recours à des services offerts par un autre éditeur dans le but de répondre à un marché spécifique qu'il ne peut pas couvrir avec ses propres services. On parle alors d'entreprise étendue (voir figure 5.3b) [Fakhfakh et al., 2010].

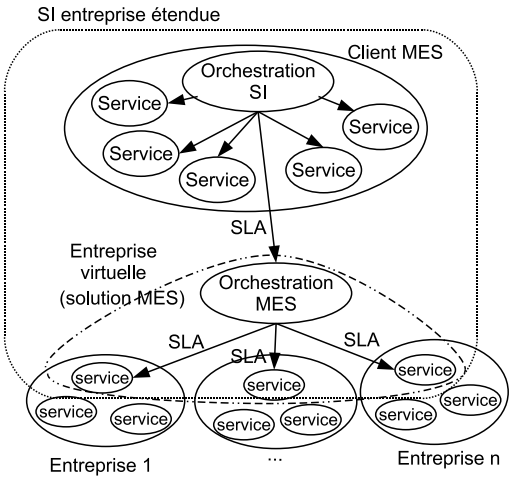


(a) SI d'une entreprise virtuelle

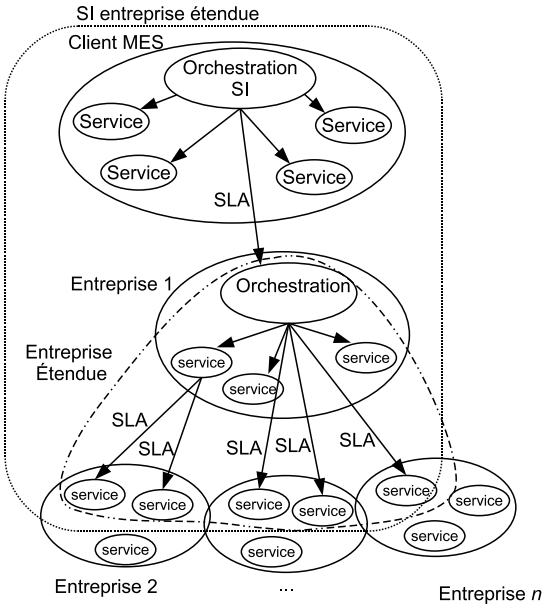
(b) SI d'une entreprise étendue

Figure 5.3 – Exemples d'architectures orientées services pour des systèmes d'information inter-entreprises

La solution MES, ou plus généralement, le système d'information MES, est amené à



(a) Entreprise cliente intégrant une entreprise virtuelle



(b) Entreprise cliente intégrant une entreprise étendue

Figure 5.4 – Exemples d'architectures basées services pour un système d'information intégrant une solution MES

être intégré dans une entreprise cliente MES. Là encore, nous pouvons distinguer différentes situations selon que le système d'information MES suit l'architecture d'une entreprise virtuelle ou étendue. La première situation concerne par exemple un client MES qui souhaite intégrer une solution MES plus ou moins complète. Nous nous retrouvons donc dans le cas (voir figure 5.4a) d'une entreprise étendue (entreprise cliente MES) qui intègre un système d'information d'une entreprise virtuelle (solution MES). La deuxième situation représente par exemple le cas d'une entreprise cliente MES qui possède certains services MES et qui envisage d'étendre sa couverture fonctionnelle du MES avec des services de certains des partenaires du consortium. C'est le cas d'une entreprise étendue (l'entreprise cliente MES) qui intègre des services d'une autre entreprise elle-même étendue (voir figure 5.4b). Une troisième situation est envisageable également lorsque l'entreprise cliente MES souhaite intégrer indépendamment certains services offerts par certains éditeurs. Cette situation est exactement le cas d'une entreprise étendue (voir figure 5.3b).

5.2.5 Synthèse

Nous avons présenté dans cette section la partie de nos travaux qui a été réalisée dans le cadre du projet MES (section 5.2.1). Ces travaux concernent le transfert scientifique aux industriels, plus précisément l'application des architectures orientées services dans le domaine du MES. Dans ce sens, nous avons contribué dans le cadre du projet MES, à la définition de l'architecture de la solution logicielle MES (section 5.2.2) comme une architecture orientée services. Ceci est concrétisé via l'élaboration d'une cartographie de

services MES (section 5.2.3) qui comprend 242 opérations de service regroupées en 23 services MES. L'ensemble des services est conforme aux standards du Web 2.0 et à la norme ISA-95, spécifique au domaine du MES. Cette cartographie de services MES constitue un catalogue de fonctionnalités. Ainsi, les clients MES auront la flexibilité de choisir les services et les opérations de services qui les intéressent et de les orchestrer de la manière qui leur convient. L'implémentation de ces travaux a donné naissance à un nouveau produit, MESTRIA, représentant une solution MES agile au sens où sa couverture fonctionnelle du MES est variable selon les demandes des clients.

D'autre part, nous avons étudié et proposé plusieurs cas de figures d'architectures de systèmes d'information intégrant la solution MES (section 5.2.4) pour lesquels différentes architectures d'entreprises virtuelles et étendues peuvent être envisagées. Le choix d'une architecture plutôt qu'une autre a des conséquences en termes de ressources, de maîtrise, d'organisation (technologique et managériale), de relations inter-entreprises (sous-traitance, co-traitance, etc.), etc. La gestion de la qualité de service, en particulier des contrats (SLA), est aussi impactée par le choix de l'architecture. Rappelons que les contrat (SLA) précisent entre autres, les niveaux de qualité des services (les exigences en terme de qualité) ainsi que leurs conditions d'invocation. À titre d'exemple, dans le cas des architectures données dans la figure 5.4, le client MES n'a besoin d'établir qu'un seul contrat (SLA) avec le fournisseur de la solution MES qui est *(i)* l'entreprise virtuelle dans le cas de l'architecture donnée en figure 5.4a ou *(ii)* l'entreprise étendue dans le cas de l'architecture donnée en figure 5.4b. Par contre, dans le cas de l'architecture d'une entreprise étendue (voir figure 5.3b), le client MES est amené à négocier des contrats (SLA) avec chacun des partenaires offrant des services. Ces contrats (SLA) doivent être surveillés afin de vérifier le respect des conditions d'utilisation (du côté client) et des garanties agréées sur les attributs qualité (du côté fournisseur). En particulier, la supervision de la qualité des orchestrations de services est importante pour assurer et garantir le bon fonctionnement des processus métiers et de l'entreprise en général. C'est dans ce contexte que nous avons proposé notre approche pour la supervision des orchestrations de services. Nous nous intéressons dans la suite, à l'application de notre approche de supervision des orchestrations de services. Pour cela, nous commençons par présenter dans la section suivante un exemple d'orchestration de services supportant un processus métier MES.

5.3 Exemple d'orchestration de services MES

Plusieurs scénarios MES ont été proposés dans le cadre du projet MES et sont aujourd'hui gérés par la solution MESTRIA. Dans cette section, nous choisissons un scénario parmi les scénarios MES afin d'illustrer ensuite notre approche de supervision. Le scénario que nous avons choisi, décrit un processus de production à l'affaire dont nous ne présentons qu'un extrait ci-après pour des raisons de simplification. Le modèle de l'orchestration de ce processus métier est présenté en figure 5.5. Ce modèle comporte quatorze invocations d'opérations de service issues de la cartographie de services MES [Fakhfakh et al., 2011c].

Ces opérations de service sont données dans le tableau 5.1.

Le lancement de l'affaire commence par la création des données techniques (S_1) provenant de l'ERP dans la solution MES. À titre d'exemple, cette fonctionnalité métier est assurée par l'opération de service *DefineProductionRequest* appartenant au service MES *Production Requests Management*. Ensuite, l'affaire est ordonnancée (S_2) puis, une maintenance préventive est réalisée (S_3) en même temps que le lancement en production (S_4). Ceci est décrit à l'aide du patron de *workflow Parallel Split*. Notons que l'affaire peut contenir plusieurs opérations de production dans sa gamme de fabrication. Ces opérations de production peuvent avoir été ordonnancées pour être réalisées en parallèle, en séquence, etc. Afin de simplifier, nous supposons ici que les opérations de production sont réalisées en séquence. Pour chacune de ces opérations de production, un processus de traçabilité est effectué (S_5 à S_8). Ceci est donc modélisé via le patron boucle. Notons que dans ce cas, le nombre d'itérations du patron boucle est inconnu avant le lancement de l'affaire (pendant la phase de conception) puisqu'on ne connaît pas à ce moment là, le nombre d'opérations de production liées à l'affaire. Cependant, dès que l'affaire est créée, le nombre d'opérations de production et par suite le nombre d'itérations l, est parfaitement connu. Le processus de traçabilité consiste à tracer les matières et les produits semi-finis consommés (S_5), suivre la main d'œuvre (S_6), suivre la production (S_7), et enfin tracer *(i)* les produits semi-finis issus des opérations de production intermédiaires ou *(ii)* le produit fini dans le cas de la dernière opération de production de l'affaire (S_8). Une fois que la production est finie, un contrôle qualité final est réalisé sur le produit (S_9). Le produit est ensuite mis en stock (S_{10}) pour être expédié par la suite (S_{11}). À ce moment, l'affaire peut être terminée (S_{12}). Enfin, les documents relatifs à la production et à la maintenance sont récupérés (S_{13}) pour être enregistrés dans la Gestion Électronique des Documents (GED) (S_{14}). Notons que les documents relatifs à la maintenance et à la production sont traités indépendamment les uns des autres. Autrement dit, la récupération et l'enregistrement des documents de la maintenance n'est pas forcément liée à la fin de l'affaire en production. D'où l'utilisation du patron de *workflow Multi-Merge* (voir section 3.3.2) pour la convergence des branches en parallèle (voir figure 5.5).

Ainsi, nous avons présenté un exemple d'orchestration de services supportant un processus métier MES. Dans la section suivante, nous appliquons notre approche de supervision sur ce modèle d'orchestration et nous montrons les avantages qu'elle présente par rapport aux travaux existants.

5.4 Mise en œuvre de l'approche de supervision : phase de conception

La mise en œuvre de notre approche de supervision commence par une étape de configuration lors de la phase de conception, où les paramètres de réglage sont fixés et les données nécessaires à l'approche sont établies (voir figure 3.1). Ceci inclut :

- les données relatives aux attributs qualité des services impliqués dans l'orchestration.

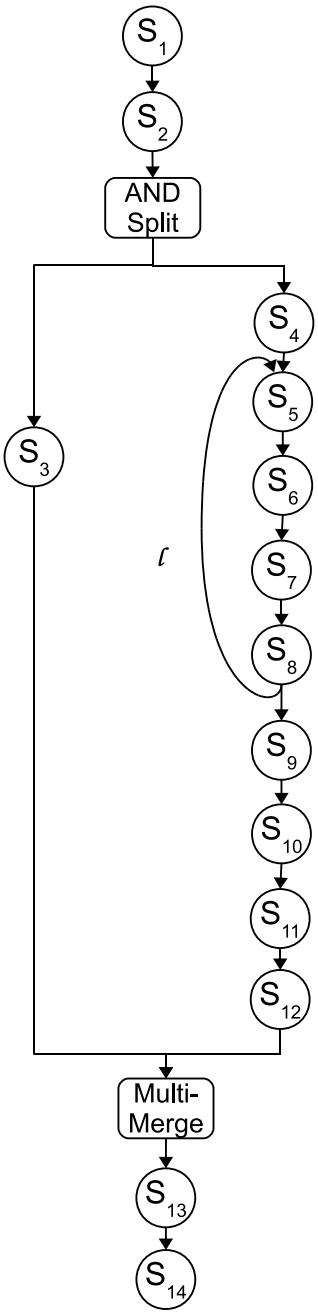


Figure 5.5 – Extrait d'un modèle d'orchestration d'un processus de production à l'affaire

Tableau 5.1 – Description des activités d'un exemple de processus métier MES

Activité	**Fonctionnalité métier**	***Opération de service***	***Service***
S_1	Créer les données techniques de l'affaire	*DefineProduction-Request*	*Production Requests Management*
S_2	Ordonnancer l'affaire	*RefreshPlanning*	*Production Planning*
S_3	Réaliser maintenance préventive	*CreateMaintenance-WorkOrder*	*Maintenance Work Order Management*
S_4	Lancer en production	*UpdateProduction-RequestState*	*Production Requests Management*
S_5	Tracer matières premières et produits semi-finis	*ReportConsumption*	*Production Tracking*
S_6	Suivre la main d'œuvre	*TrackRessource*	*Production Tracking*
S_7	Suivre la production	*TrackRessource*	*Production Tracking*
S_8	Tracer produit finis et semi-fini	*ReportProduction*	*Production Tracking*
S_9	Contrôler le produit final	*LoadInspection-Specification*	*Quality Inspection Management*
S_{10}	Mettre en stock	*AddToStock*	*Inventory Movement Management*
S_{11}	Expédier produit fini	*CreateDeliveryOrder*	*Inventory Delivery Management*
S_{12}	Production terminée	*UpdateProduction-RequestState*	*Production Requests Management*
S_{13}	Récupérer documents	*GetDocument*	*Enterprise Content Management*
S_{14}	Enregistrer documents	*ReportDocument*	*Enterprise Content Management*

Cela implique de définir les valeurs « théoriques » des attributs qualité déterministes ainsi que les distributions de probabilités des attributs qualité non déterministes ;
- les données relatives au modèle d'orchestration, notamment les probabilités des branches conditionnelles et les informations concernant le nombre d'itérations dans les patrons de boucle, s'ils existent ;
- le choix de la stratégie de surveillance (réactive ou préventive) ainsi que le seuil de probabilité lié aux conditions de déclenchement de l'adaptation correspondant à cette stratégie. Dans le cas où la stratégie de surveillance préventive est choisie (par le client), il faut fixer le paramètre de réglage β (voir expression 4.29, page 120) de la condition de déclenchement de l'adaptation dynamique ;
- les exigences du client en terme de qualité de l'orchestration, plus précisément le niveau de qualité désiré et le niveau de qualité minimum acceptable qui sont nécessaires pour la construction du modèle de préférences ;
- les préférences et les intensités de préférences du client pour construire le modèle de préférences.

En ce qui concerne les données relatives aux attributs qualité, nous allons considérer des distributions de probabilités « fictives » pour les temps de réponse relatifs aux opérations de services, tandis que pour la fiabilité et la disponibilité nous prenons des valeurs déterministes (fixes) (voir tableau 5.2). À titre d'exemple, les temps de réponse de chacune des opérations de service sont représentés par des distributions gaussiennes (définies par leur moyenne μ et leur écart type σ). Notons que grâce à la discrétisation (voir section 3.4.1), nous pouvons prendre en compte n'importe quel type de distribution de probabilités. En pratique, ces données relatives aux attributs qualité sont fournies potentiellement par les fournisseurs des services ou déduites à partir des historiques d'exécutions des services. Les probabilités dans les patrons de choix (s'ils existent) sont soient déduites à partir d'un historique d'exécution de l'orchestration ou du processus métier, soient déduites de la loi d'équiprobabilité. En ce qui concerne le nombre d'itérations dans le patron boucle, il peut être soit déterministe lorsqu'il s'agit d'un nombre fixe l d'itérations, soit aléatoire lorsqu'il est inconnu. Dans l'exemple du processus MES que nous avons pris, le nombre l représente le nombre d'opérations de production liées à l'affaire en production. Ce nombre est inconnu jusqu'à la création et la définition de l'affaire. Par conséquent, nous allons représenter le nombre d'itérations, pendant la phase de conception, par une variable aléatoire L dont le domaine est $Dom(L) = \{1, 2, 3\}$ et la loi de probabilité est $P_L = \{0.4, 0.4, 0.2\}$.

D'autre part, nous considérons dans la suite, la stratégie de surveillance réactive avec un seuil de probabilité s égal à 20%. Nous supposons aussi que les exigences spécifiées par le client pour le processus métier considéré, sont les suivantes (voir section 4.2.2, page 91) :

$$QoSL^{good}(orch) = (q_{rt}^{good}, q_{rel}^{good}, q_{av}^{good}) = (50, 1, 1) \tag{5.1}$$

$$QoSL^{neutral}(orch) = (q_{rt}^{neutral}, q_{rel}^{neutral}, q_{av}^{neutral}) = (70, 0.2, 0.2) \tag{5.2}$$

Tableau 5.2 – Données des attributs qualité relatives aux opérations de service du scénario MES

Activité	**Temps de réponse (min)**		**Fiabilité**	**Disponibilité**
	μ	σ		
S_1	2	0.2	0.95	0.97
S_2	2	0.2	0.9	0.95
S_3	30	3	0.95	0.95
S_4	1	0.1	0.95	0.97
S_5	1	0.1	0.97	0.96
S_6	1	0.1	0.97	0.96
S_7	1	0.1	0.97	0.96
S_8	1	0.1	0.97	0.96
S_9	15	1.5	0.95	0.99
S_{10}	5	0.5	0.97	0.95
S_{11}	20	2	0.97	0.95
S_{12}	1	0.1	0.95	0.95
S_{13}	1	0.1	0.9	0.97
S_{14}	1	0.1	0.9	0.97

μ : moyenne
σ : écart-type

En pratique, ces exigences peuvent être déjà formalisées directement dans le contrat (SLA) dans le cas où il y a un seul contrat établi sur l'orchestration globale (voir par exemple les architectures données en figure 5.4). Dans le cas où le client établit un contrat (SLA) par service impliqué dans l'orchestration (c'est le cas par exemple de l'architecture donnée en figure 5.3b), les contraintes peuvent être agrégées moyennant les règles d'agrégation de patrons de *workflow* et il en résulte ainsi une contrainte globale par attribut qualité (voir section 2.4.1).

Rappelons que le modèle de préférences du client sert à évaluer la qualité globale de l'orchestration au regard des exigences spécifiées par le client. C'est sur la base de ce modèle de préférences que s'appuie la surveillance de la qualité globale de l'orchestration et la gestion de son adaptation dynamique. Dans la section suivante, nous allons construire un modèle de préférences relatif au processus métier décrit dans la section 5.3. Ce modèle de préférences, une fois construit, reste valable pour tout le cycle de vie de l'orchestration correspondant au processus métier ainsi que pour autres modèles d'orchestrations

Tableau 5.3 – Exemple de niveaux de qualité pour la construction des fonctions d'utilité marginales

Niveaux de qualité	**Temps de réponse**	**Fiabilité**	**Disponibilité**
$QoSL_1 = QoSL^{good}$	50	1	1
$QoSL_2$	54	0.4	0.9
$QoSL_3$	58	0.8	0.6
$QoSL_4$	60	0.7	0.7
$QoSL_5$	62	0.1	0.95
$QoSL_6$	64	0.6	0.8
$QoSL_7$	66	0.85	0.85
$QoSL_8 = QoSL^{neutral}$	70	0.2	0.2
$QoSL_9$	74	0.9	0.4
$QoSL_{10}$	80	0.95	0.1

susceptibles de correspondre au même processus métier.

5.4.1 Construction du modèle de préférences

Nous avons vu dans la section 4.3 que la construction du modèle de préférences du client consiste en deux étapes :

- la construction des fonctions d'utilité (normalisation) u_{rt}, u_{rel} et u_{av} relatives aux attributs qualité retenus, à savoir le temps de réponse (q_{rt}), la fiabilité (q_{rel}) et la disponibilité (q_{av}).
- la construction de la fonction d'agrégation définie par l'opérateur de l'intégrale de Choquet 2-additive. Ceci revient à déterminer les paramètres qui interviennent dans cet opérateur d'agrégation, à savoir les trois indices de *Shapley* (poids d'importance) ν_{rt}, ν_{rel} et ν_{av} ainsi que les trois paramètres d'interaction I_{rt-rel}, I_{rt-av} et I_{rel-av}.

Pour arriver à cette fin, nous avons besoin de quatre matrices de préférences remplies par le client (trois matrices pour la construction de chacune des fonctions d'utilité marginales et une matrice pour la construction de la fonction d'utilité globale basée sur l'intégrale de Choquet 2-additive). Ces matrices sont remplies lors d'une interview réalisée avec le client. En ce qui concerne les matrices de préférences dédiées à la construction des fonctions d'utilité marginales, nous avons besoin d'un certain nombre γ ($\gamma \in \mathbb{N}$) de situations (niveaux de qualité) pour qu'elles soient comparées (par le client). Ce nombre γ détermine la précision de la normalisation ; plus il est grand, meilleure est la précision (voir ci-après). Notons que la question ne se pose pas pour la construction de la fonction d'utilité globale. Dans la suite, nous allons considérer dix niveaux de qualité ($\gamma = 10$) pour la construction des fonctions d'utilité marginales. Ces niveaux de qualité doivent couvrir la zone de tolérance (c'est-à-dire la plage de valeurs entre le niveau de qualité désiré et le niveau de qualité minimum acceptable), en particulier le domaine de définition des variables aléatoires lors-

Tableau 5.4 – Matrice de préférences pour le temps de réponse

q_{rt}	$QoSL_1$	$QoSL_2$	$QoSL_3$	$QoSL_4$	$QoSL_5$	$QoSL_6$	$QoSL_7$	$QoSL_8$	$QoSL_9$	$QoSL_{10}$
$QoSL_1$	nulle	3	P	P	P	P	P	P	P	P
$QoSL_2$		nulle	2	P	P	P	P	P	P	P
$QoSL_3$			nulle	1	P	P	P	P	P	P
$QoSL_4$				nulle	2	P	P	P	P	P
$QoSL_5$					nulle	1	P	P	P	P
$QoSL_6$						nulle	2	P	P	P
$QoSL_7$							nulle	1	P	P
$QoSL_8$								nulle	3	P
$QoSL_9$									nulle	4
$QoSL_{10}$										nulle

$0 = nulle,\ 1 = très\,faible,\ 2 = faible,\ 3 = modérée,\ 4 = fort,\ 5 = très\,fort,\ 6 = extrême$

qu'on a des attributs qualité non déterministes). À titre d'exemple, nous allons prendre les niveaux de qualité présentés dans le tableau 5.3 pour la construction des fonctions d'utilité. Ces niveaux de qualité sont présentés au client pour qu'il exprime ses préférences ainsi que ses intensités de préférence comme nous l'avons présenté dans la section 4.3.2. Les matrices de préférences relatives à chacun des attributs qualité sont données dans les tableaux 5.4, 5.5 et 5.6. Prenons l'exemple de la matrice de préférences relative au temps de réponse. Rappelons que la construction de la fonction d'utilité marginale u_{rt} consiste à construire les valeurs $u_{rt}(q^i_{rt})$ correspondant à chaque valeur du temps de réponse q_{rt} d'un niveau de qualité i ($QoSL_i$). Sachant que $u_{rt}(q^1_{rt}) = u_{rt}(q^{good}_{rt}) = 1$ et $u_{rt}(q^8_{rt}) = u_{rt}(q^{neutral}_{rt}) = 0$, les inconnues restant sont $u_{rt}(q^2_{rt})$, $u_{rt}(q^3_{rt})$, $u_{rt}(q^4_{rt})$, $u_{rt}(q^5_{rt})$, $u_{rt}(q^6_{rt})$, $u_{rt}(q^7_{rt})$, $u_{rt}(q^9_{rt})$, $u_{rt}(q^{10}_{rt})$ et α. Le système d'équations, déduit de la matrice donnée en tableau 5.4, mis sous la forme matricielle $A_{rt} \times X_{rt} = B_{rt}$ est :

$$\underbrace{\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -4 \end{bmatrix}}_{A_{rt}} \times \underbrace{\begin{bmatrix} u_{rt}(q^2_{rt}) \\ u_{rt}(q^3_{rt}) \\ u_{rt}(q^4_{rt}) \\ u_{rt}(q^5_{rt}) \\ u_{rt}(q^6_{rt}) \\ u_{rt}(q^7_{rt}) \\ u_{rt}(q^9_{rt}) \\ u_{rt}(q^{10}_{rt}) \\ \alpha \end{bmatrix}}_{X_{rt}} = \underbrace{\begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{B_{rt}} \tag{5.3}$$

Tableau 5.5 – Matrice de préférences pour la fiabilité

q_{rel}	$QoSL_1$	$QoSL_{10}$	$QoSL_9$	$QoSL_7$	$QoSL_3$	$QoSL_4$	$QoSL_6$	$QoSL_2$	$QoSL_8$	$QoSL_5$
$QoSL_1$	nulle	1	P	P	P	P	P	P	P	P
$QoSL_{10}$		nulle	2	P	P	P	P	P	P	P
$QoSL_9$			nulle	3	P	P	P	P	P	P
$QoSL_7$				nulle	1	P	P	P	P	P
$QoSL_3$					nulle	4	P	P	P	P
$QoSL_4$						nulle	2	P	P	P
$QoSL_6$							nulle	1	P	P
$QoSL_2$								nulle	2	P
$QoSL_8$									nulle	1
$QoSL_5$										nulle

$0 = nulle,\ 1 = très\, faible,\ 2 = faible,\ 3 = modérée,\ 4 = fort,\ 5 = très\, fort,\ 6 = extrême$

Tableau 5.6 – Matrice de préférences pour la disponibilité

q_{rel}	$QoSL_1$	$QoSL_5$	$QoSL_2$	$QoSL_7$	$QoSL_6$	$QoSL_4$	$QoSL_3$	$QoSL_9$	$QoSL_8$	$QoSL_{10}$
$QoSL_1$	nulle	1	P	P	P	P	P	P	P	P
$QoSL_5$		nulle	1	P	P	P	P	P	P	P
$QoSL_2$			nulle	2	P	P	P	P	P	P
$QoSL_7$				nulle	3	P	P	P	P	P
$QoSL_6$					nulle	2	P	P	P	P
$QoSL_4$						nulle	1	P	P	P
$QoSL_3$							nulle	2	P	P
$QoSL_9$								nulle	2	P
$QoSL_8$									nulle	2
$QoSL_{10}$										nulle

$0 = nulle,\ 1 = très\, faible,\ 2 = faible,\ 3 = modérée,\ 4 = fort,\ 5 = très\, fort,\ 6 = extrême$

Ce système d'équations admet une unique solution qui est :

$$X_{rt}^{t} = [0.75\ ;\ 0.58\ ;\ 0.50\ ;\ 0.33\ ;\ 0.25\ ;\ 0.08\ ;\ -0.25\ ;\ -0.58\ ;\ 0.08] \tag{5.4}$$

Nous pouvons observer par exemple que les niveaux de qualité $QoSL_9$ et $QoSL_{10}$ violent les exigences du client en ce qui concerne le temps de réponse ($u_{rt}(q_{rt}^{9}) < 0$, $u_{rt}(q_{rt}^{10}) < 0$). La même procédure est réalisée pour construire les fonctions d'utilité marginales relatives à la fiabilité et à la disponibilité. Ainsi, nous obtenons :

$$X_{rel}^{t} = [0.13\ ;\ 0.56\ ;\ 0.31\ ;\ -0.06\ ;\ 0.19\ ;\ 0.63\ ;\ 0.81\ ;\ 0.94\ ;\ 0.06] \tag{5.5}$$

$$X_{av}^{t} = [0.86\ ;\ 0.29\ ;\ 0.36\ ;\ 0.93\ ;\ 0.5\ ;\ 0.71\ ;\ 0.14\ ;\ -0.14\ ;\ 0.07] \tag{5.6}$$

La figure 5.6 montre une comparaison des tracés des trois fonctions d'utilité marginales selon que l'on applique la méthode que nous avons utilisée dans notre approche (c'est-à-dire moyennant l'expression des préférences et des intensités de préférences), ou que l'on utilise des équations linéaires [Taher et al., 2005b, Herssens et al., 2008]. Les trois fonctions d'utilité marginales construites précédemment sont tracées en ligne continue (—). Rappelons que le nombre de niveaux de qualité considéré pour la construction de ces fonctions d'utilité marginales est égal à dix ($\gamma = 10$). Nous avons modifié le nombre de niveaux de qualité à huit ($\gamma = 8$) et tracé les courbes correspondantes aux trois fonctions d'utilité marginales (ligne $-*-$). Le résultat (voir figure 5.6) montre que les fonctions d'utilité marginales relatives à la fiabilité et à la disponibilité sont légèrement moins précises que celles où nous avons considéré dix niveaux de qualité, alors que les fonctions d'utilité marginales du temps de réponse sont presque identiques dans les deux cas. Cela montre l'influence du nombre de niveaux de qualité considéré dans la normalisation sur la précision des fonctions d'utilité marginales. D'autre part, nous constatons une différence significative entre la normalisation basée sur les préférences (fournies par le client), et celle où l'on utilise des équations linéaires (courbes en ligne discontinue - -), en particulier en ce qui concerne la fiabilité et la disponibilité. À titre d'exemple, du point de vue de la disponibilité, la satisfaction du client est linéaire pour les mauvaises et les bonnes valeurs tandis qu'entre les deux la linéarisation sur-estime la satisfaction du client. Le cas de la fiabilité présente un mixte de sous-estimation (pour $q_{av} < 0.2$) et de sur-estimation (pour $0.2 < q_{rel} < 0.95$) de la satisfaction du client ; cette dernière étant linéaire pour $q_{rel} > 0.95$. Enfin, nous pouvons considérer que la satisfaction du client vis-à-vis du temps de réponse est quasi-linéaire. En guise de conclusion, nous estimons que la méthode de normalisation que nous avons utilisée dans notre approche, est meilleure en terme de précision (que celle où l'on utilise des équations linéaires) du fait qu'elle représente mieux la satisfaction du client.

Les équations linéaires utilisées dans le tracé donné en figure 5.6 sont issues de [Taher et al., 2005b] :

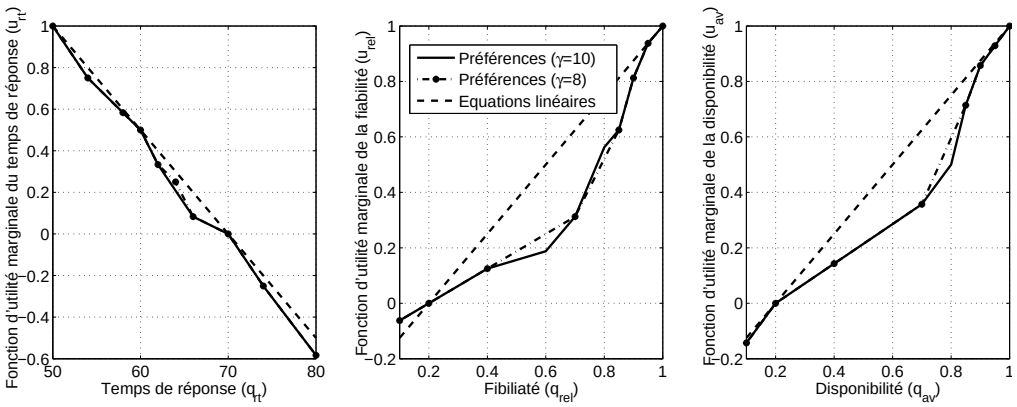


Figure 5.6 – Normalisation des attributs qualité

$$u_i(q_i^j) = \frac{q_i^{neutral} - q_i^j}{q_i^{neutral} - q_i^{good}} \tag{5.7}$$

$$u_i(q_i^j) = \frac{q_i^j - q_i^{neutral}}{q_i^{good} - q_i^{neutral}} \tag{5.8}$$

où l'équation 5.7 est utilisée pour normaliser les attributs qualité de dimension descendante comme le temps de réponse, et l'équation 5.8 concerne les attributs qualité de dimension ascendante comme la fiabilité et la disponibilité dans notre exemple. Notons que dans la méthode que nous avons présentée précédemment, nous construisons un ensemble fini de valeurs des fonctions d'utilité marginales. Les fonctions d'utilité complètes (permettant de normaliser n'importe quelle valeur du domaine de définition des attributs qualité) sont obtenues par interpolation linéaire (voir équation 4.6, page 98) comme le montre le tracé présenté en figure 5.6.

Les fonctions d'utilité marginales que nous avons construites précédemment représentent la première étape dans la construction du modèle de préférences du client. La deuxième étape consiste en la construction de la fonction d'utilité globale basée sur l'opérateur de l'intégrale de Choquet 2-additive (voir section 4.4). Cet opérateur d'agrégation est entièrement défini par les indices de *Shapley* et les paramètres d'interaction. Dans le cas de trois attributs qualité, ces paramètres sont au nombre de six : trois indices de *Shapley* (poids d'importance) ν_{rt}, ν_{rel} et ν_{av}, et trois paramètres d'interaction I_{rt-rel}, I_{rt-av} et I_{rel-av}. Afin de déterminer ces paramètres, le client est amené à remplir la matrice de préférences constituée des niveaux de qualité binaires (voir section 4.4.3). À titre d'exemple, considérons la matrice de préférences donnée en tableau 5.7. La forme matricielle du système d'équations extraites de cette matrice de préférences est identique à celle donnée dans l'expression 4.19 (page 111) en remplaçant les intensités de préférences h_i, $1 \leq i \leq 7$, par les valeurs numériques données dans la matrice des préférences du tableau 5.7. La résolution

Tableau 5.7 – Matrice de préférence pour la détermination des paramètres de l'intégrale de Choquet 2-additive

	(1,1,1)	(0,1,1)	(1,1,0)	(0,1,0)	(1,0,1)	(1,0,0)	(0,0,1)	(0,0,0)
(1,1,1)	No	2	P	P	P	P	P	P
(0,1,1)		No	4	P	P	P	P	P
(1,1,0)			No	2	P	P	P	P
(0,1,0)				No	2	P	P	P
(1,0,1)					No	3	P	P
(1,0,0)						No	3	P
(0,0,1)							No	1

de ce système d'équations nous fournit les valeurs suivantes des paramètres :

$$\nu_{rt} = 0.21 \quad \nu_{rel} = 0.56 \quad \nu_{av} = 0.23 \quad I_{rt-rel} = -0.18 \quad I_{rt-av} = 0.06 \quad I_{rel-av} = 0.23 \ . \tag{5.9}$$

Nous pouvons vérifier que les conditions de monotonie (voir expression 4.13, page 108) sont respectées :

$$\nu_{rt} - \frac{1}{2}(|I_{rt-rel}| + |I_{rt-av}|) = 0.21 - 0.5(0.18 + 0.06) \geq 0 \tag{5.10}$$

$$\nu_{rel} - \frac{1}{2}(|I_{rt-rel}| + |I_{rel-av}|) = 0.56 - 0.5(0.18 + 0.23) \geq 0 \tag{5.11}$$

$$\nu_{av} - \frac{1}{2}(|I_{rt-av}| + |I_{rel-av}|) = 0.23 - 0.5(0.06 + 0.23) \geq 0 \ . \tag{5.12}$$

De cette manière, la fonction d'utilité globale du modèle de préférences est entièrement définie et par suite la construction du modèle de préférences du client est accomplie. Nous avons montré à travers un exemple dans le chapitre précédent (voir section 4.4) que l'intégrale de Choquet permet de prendre en compte les dépendances préférentielles que peut manifester le client dans l'expression de ses préférences. Nous allons maintenant établir une comparaison entre le modèle de préférences que nous utilisons dans notre approche et d'autres modèles existants dans la littérature [Zeng et al., 2004, Taher et al., 2005b, Canfora et al., 2008, Herssens et al., 2008]. Pour cela, nous allons considérer [Fakhfakh et al., 2012] :

1. un modèle de préférences basé sur l'opérateur de la moyenne pondérée et la normalisation par des équations linéaires [Zeng et al., 2004, Taher et al., 2005b, Canfora et al., 2008] ;
2. un modèle de préférences basé sur l'opérateur de la moyenne pondérée et la normalisation par l'expression de préférences. Ce modèle correspond au cas où nous utilisons la méthode MACBETH basée sur la moyenne pondérée [Fakhfakh et al., 2011f] ;
3. un modèle de préférence basé sur l'opérateur de l'intégrale de Choquet 2-additive et la normalisation par des équations linéaires [Herssens et al., 2008] ;

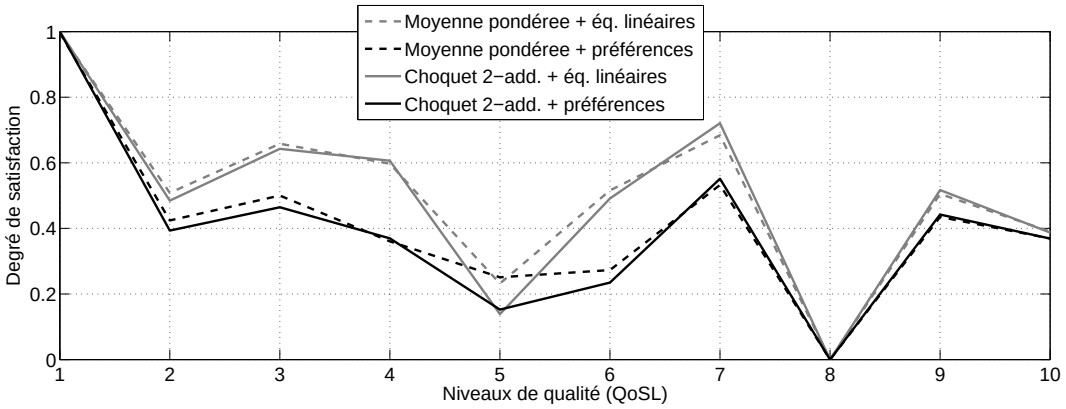


Figure 5.7 – Comparaison entre différents modèles de préférences

4. notre modèle de préférences basé sur l'opérateur de l'intégrale de Choquet 2-additive et la normalisation par l'expression de préférences [Fakhfakh et al., 2011d].

La comparaison est effectuée par rapport aux degrés de satisfaction des dix niveaux de qualité [3] présentés dans le tableau 5.3. Le résultat est donnée en figure 5.7. Le tracé du modèle (1) (respectivement du modèle (2)) est représenté par la courbe en ligne noire discontinue (respectivement en ligne grise discontinue), tandis que le tracé du modèle (3) (respectivement du modèle (4)) est représenté par la courbe en ligne grise continue (respectivement en ligne noire continue). La première constatation que nous pouvons dégager, concerne le niveau de qualité 5 ($QoSL_5$) où les modèles (1) et (2), basés sur la moyenne pondérée, sur-estiment la satisfaction du client par rapport aux modèles (3) et (4), basés sur l'intégrale de Choquet 2-additive. Ceci est très intéressant car cela montre l'impact des dépendances préférentielles sur le degré de satisfaction global de l'orchestration. En effet, les degrés de satisfaction fournis respectivement par les modèles basés sur la moyenne pondérée et par ceux basés sur l'intégrale de Choquet 2-additive, sont situés de part et d'autre du degré de satisfaction « 0.2 ». Imaginons que le degré de satisfaction « 0.2 » corresponde au seuil de déclenchement de l'adaptation dynamique (selon une stratégie de surveillance préventive par exemple) dans un système de supervision, ou encore que les exigences du client (plus précisément le niveau de qualité minimum acceptable $QoSL^{neutral}$) fassent en sorte que le « 0 » soit à la place de « 0.2 ». Dans ce cas, le système de supervision exploitant un modèle basé sur la moyenne pondérée ne détecte pas ce dépassement de seuil alors qu'il le devrait selon les modèles basés sur l'intégrale de Choquet 2-additive. Par ailleurs, les degrés de satisfaction correspondant aux autres niveaux de qualité (par exemple 3, 4, 6, 7, etc) consolident ce que nous avons mentionné précédemment à propos de l'impact de la méthode de normalisation (soit par l'expression des préférences, soit par les équations linéaires, voir figure 5.6) sur la représentation de la satisfaction du client (degré de

3. Notons que les travaux de [Canfora et al., 2008] ne sont pas concernés par les analyses relatives aux niveaux de qualité 5, 9 et 10. En effet, vu que certains attributs qualité de ces niveaux de qualité violent certaines contraintes globales ($q_i^{neutral}$), la fonction d'utilité globale dans ces travaux (voir expression 2.1, page 40) ne donne pas dans ce cas le même résultat que la moyenne pondérée. Cependant, pour les autres niveaux de qualité, le degré de satisfaction correspond se ramène à celui fourni par la moyenne pondérée.

satisfaction). Prenons le cas des deux courbes en ligne continue (respectivement en ligne discontinue) (voir figure 5.7). Bien que ces deux courbes correspondent aux modèles basés sur le même opérateur d'agrégation qu'est l'intégrale de Choquet 2-additive (respectivement la moyenne pondérée), nous constatons une différence, non négligeable dans certains cas, due à la méthode de normalisation. Cette différence traduit une sur-estimation de la satisfaction du client des modèles basés sur la normalisation linéaire par rapport à ceux basés sur les préférences du client. Ceci bien entendu, sera répercuté sur le système de supervision et peut résulter en une mauvaise gestion de la satisfaction du client, où l'on estime que la satisfaction du client est assurée alors que ce n'est pas le cas réellement.

Au terme de l'analyse, nous estimons que notre modèle de préférences présente une meilleure précision dans la représentation de la satisfaction du client. Ceci découle de la prise en compte, non seulement des dépendances préférentielles dans la construction de la fonction d'agrégation, mais aussi des préférences du client dans la construction des fonctions d'utilité (normalisation).

Après avoir fixé les paramètres de réglage (choix de la stratégie de surveillance, fixation du seuil de probabilité s, etc) et validé le modèle de préférences, l'approche globale d'agrégation ainsi que la gestion du degré de satisfaction de l'orchestration peuvent être appliquées. Dans la section suivante, nous nous focalisons sur l'application de l'approche d'agrégation globale pendant la phase de conception et montrons ses avantages et son intérêt.

5.4.2 Application de l'approche d'agrégation pendant la phase de conception

Rappelons que notre approche d'agrégation consiste en deux phases (voir figure 3.2), l'agrégation par application des règles de patrons de *workflow* et l'agrégation basée sur une méthode d'aide à la décision multi-critères, exploitant le modèle de préférences construit dans la section précédente.

Application des règles d'agrégation de patrons de *workflow*

Nous allons exploiter ici les règles d'agrégation de patrons de *workflow* pour obtenir chacun des attributs qualité portant sur l'orchestration globale. Ceci met en jeu les règles d'agrégation relatives aux patrons de composition qui sont utilisés dans la modélisation de l'orchestration de services correspondant au processus métier MES (voir section 5.3). Ces patrons de composition sont : séquence (PC1), boucle (PC2) et parallélisme sans synchronisation (*Multi-Merge*, PC5). Notons que les travaux que nous avons abordés dans l'état de l'art (voir section 2.4), ne permettent pas de traiter l'exemple de l'orchestration de services donné en figure 5.5. Ceci est dû à la présence du patron *Multi-Merge* (PC5) dans le modèle de l'orchestration qui ne possède pas de règle d'agrégation dans ces travaux, notamment pour le temps de réponse (voir tableau 3.2). L'agrégation est rendue possible

grâce aux règles d'agrégation que nous avons définies dans la section 3.3.3.

Pour réaliser la discrétisation du temps de réponse (voir section 3.4.1), nous avons pris vingt échantillons représentant la plage de variation du temps de réponse en se basant sur sa distribution de probabilités et ce, pour chacun des services impliqués dans l'orchestration. Nous avons ensuite appliqué la méthode d'agrégation en suivant la procédure décrite dans la section 3.3.4 pour obtenir la distribution de probabilités du temps de réponse X_{rt}, la valeur de la fiabilité q_{rel} et la valeur de la disponibilité q_{av} portant sur toute l'orchestration. Selon les données des attributs qualité relatifs aux services impliqués dans l'orchestration (voir tableau 5.2), nous obtenons le résultat suivant en ce qui concerne la fiabilité et la disponibilité :

$$q_{rel}(orch) = 0.35 \qquad\qquad q_{av}(orch) = 0.48 \ .$$

Nous avons décidé ici de mettre en place des stratégies de tolérance aux fautes afin d'améliorer ces valeurs de fiabilité et de disponibilité. À titre d'exemple, nous avons appliqué la stratégie d'invocation en boucle (voir section 4.5.3), avec un nombre de tentatives égal à deux ($n = 2$), pour les opérations des services S_2 et S_5. Ainsi, nous obtenons :

$$q_{rel}(orch) = 0.40 \qquad\qquad q_{av}(orch) = 0.54 \ .$$

La courbe représentant la fonction de répartition relative au temps de réponse global de l'orchestration, prenant en compte les stratégies de tolérance aux fautes mises en place, est donnée en figure 5.8. Rappelons que la fonction de répartition ici est obtenue par transformation inverse de la variable aléatoire discrète qui résulte de l'agrégation par les règles de patrons de *workflow* (voir section 3.4.1). Le temps de réponse moyen de l'orchestration calculé à partir de cette distribution de probabilités, est égal à 70.37 min. L'agrégation par les règles de patrons de *workflow* considérant uniquement les moyennes des temps de réponse, comme c'est le cas par exemple dans [Rosenberg, 2009,Canfora et al., 2008], fournit une moyenne de 55.71 min. L'erreur d'agrégation dans ce cas est de 21%. De surcroît, si nous interprétons ces valeurs en terme de satisfaction du client (voir tracé relatif au temps de réponse de la figure 5.6), la valeur 70.37 min est supérieure à $q_{rt}^{neutral} = 70\ min$. On est donc dans la zone de violation des exigences (insatisfaction), alors que la valeur 55.71 min est proche de $q_{rt}^{good} = 50\ min$ et donc proche de la satisfaction totale. Cette exemple montre que l'on peut avoir des situations où l'on estime que la satisfaction du client est assurée alors que réellement ce n'est pas le cas. D'où l'intérêt de représenter les attributs qualité (le temps de réponse dans notre cas) par des distributions de probabilités. L'exemple de fausse estimation de la satisfaction du client (sur-estimation dans ce cas) que nous venons de présenter, peut encore s'accentuer si le système de supervision est basé par exemple sur un modèle de préférences dont la normalisation s'effectue à l'aide des équations linéaires (modèles (1) ou (3), voir section 5.4.1). C'est typiquement le cas qu'illustrent les niveaux de qualité 3, 4, 6 et 7 (voir figure 5.7), où les modèles de préférences (1) et (3) (basés sur les équations linéaires) représentent une sur-estimation de la satisfaction totale

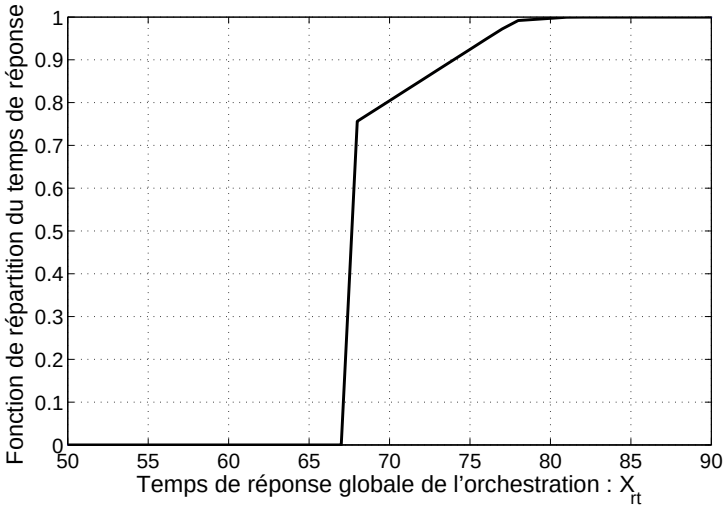


Figure 5.8 – Fonction de répartition du temps de réponse global de l'orchestration

(degré de satisfaction) par rapport aux modèles (2) et (4) (basés sur les préférences du client).

Lorsque nous avons les valeurs et les distributions de probabilités de chacun des attributs qualité considérés pour toute l'orchestration, il convient de les agréger afin d'obtenir la distribution de probabilités du degré de satisfaction de l'orchestration de services.

Application de la méthode d'aide à la décision multi-critère

Cette phase (voir figure 3.2) prend en entrée les données de la qualité des services précédemment agrégées et le modèle de préférences du client déjà construit (voir section 5.4.1). Rappelons que le modèle de préférences inclut les exigences du client et ses préférences. L'objectif de cette phase d'agrégation est de situer la qualité globale de l'orchestration par rapport aux exigences du client. Cela constitue une estimation, pendant la phase de conception, du degré de satisfaction que peut avoir l'orchestration à la fin de son exécution.

Afin d'atteindre cet objectif, nous appliquons la démarche présentée dans la section 4.5.1. La distribution de probabilités du degré de satisfaction que nous obtenons, est donnée en figure 5.9. La moyenne et l'écart-type de la distribution de probabilités du degré de satisfaction sont respectivement $\mu_{DS} = 0.10$ et $\sigma_{DS} = 0.03$. Ceci montre qu'en moyenne, la qualité globale de l'orchestration satisfait les exigences fixées par le client. Si nous examinons la condition de déclenchement de l'adaptation relative à la stratégie de surveillance réactive (voir expression 4.26), le système de supervision ne signalera pas une nécessité d'améliorer la qualité de l'orchestration (car $P_{DS}(DS < 0) = 0$, voir figure 5.9) bien que l'agrégation par les patrons de *workflow* montre qu'en moyenne le temps de réponse viole les exigences. Or, si le système de supervision se basait sur la comparaison « dure » des contraintes globales ($70.37 > 70$), comme c'est le cas dans la littérature aujourd'hui, il déclencherait l'adaptation de l'orchestration.

Toujours dans la phase de conception, cette approche d'agrégation globale permet d'envisager plusieurs cas d'utilisation. Le premier cas d'utilisation concerne la validation de la sélection des services qui a été déjà effectuée. L'analyse de la distribution de probabilités du degré de satisfaction (par exemple en calculant sa moyenne) peut être exploitée dans ce cas pour valider ou reconsidérer la sélection des services effectuée, selon que le résultat de l'analyse est favorable ou non. Un deuxième cas d'utilisation apparaît lorsque l'on a plusieurs modèles d'orchestrations de services qui supportent le même processus métier [Fakhfakh et al., 2011b]. Ces différents modèles d'orchestration peuvent avoir lieu lorsque, par exemple, il est possible d'ordonnancer les fonctionnalités métiers (réalisées par le biais des opérations de service) de différentes manières. Dans ce cas, l'analyse de la distribution de probabilités du degré de satisfaction, peut aider le client dans le choix de la bonne configuration (modèle d'orchestration) en termes de qualité. Le troisième cas d'utilisation cible la sélection des services qui vont être impliqués dans l'orchestration. Pour cela, nous pouvons exploiter dans un premier temps uniquement le modèle de préférences (construit dans la deuxième phase de notre approche globale d'agrégation) pour sélectionner chacun des meilleurs services (satisfaisant le client) parmi l'ensemble des services disponibles dans l'annuaire de services. Ce cas d'utilisation est similaire aux travaux de [Herssens et al., 2008] qui proposent une approche de sélection des services basée sur l'intégrale de Choquet 2-additive,mais avec un modèle de préférences moins précis que celui que nous proposons (voir figure 5.7). Notons que ces travaux ne considèrent pas les contraintes globales sur l'orchestration dans leur approche de sélection. Pour vérifier la satisfaction des contraintes globales (exigences), nous pouvons envisager d'appliquer dans second temps notre approche d'agrégation globale.

Les travaux de cette thèse ciblent particulièrement la supervision des orchestration de services en cours d'exécution. Dans la section suivante, nous présentons un exemple d'application dans ce contexte.

5.5 Application de l'approche de supervision au cours de l'exécution

Dans cette section, nous simulons l'exécution de l'orchestration correspondant au processus MES de la figure 5.5. Pour cela, nous commençons par présenter le contexte et les données de la simulation (section 5.5.1) avant d'illustrer l'application de l'approche d'agrégation au cours de l'exécution (section 5.5.2) et mettre en œuvre par la suite la gestion de l'adaptation dynamique (section 5.5.3).

5.5.1 Contexte de la simulation

Afin de simuler l'exécution de l'orchestration de services, nous allons générer des temps de réponse de manière aléatoire pour chaque opération de service supposée avoir été exécutée. La génération des temps de réponse se fait bien entendu en considérant la distribution

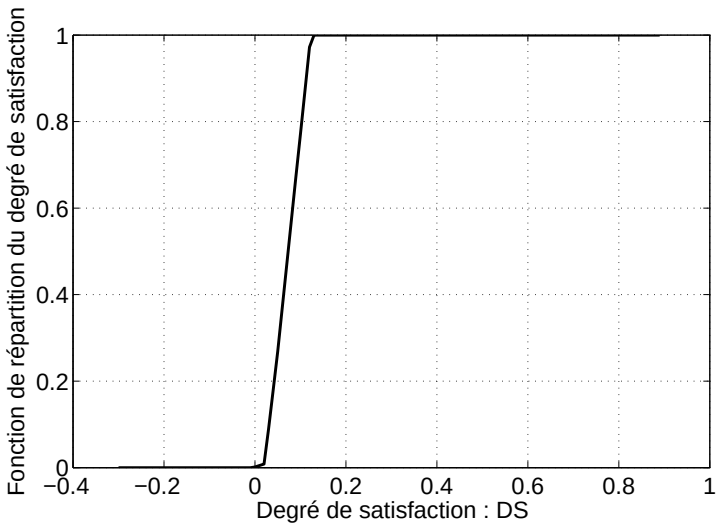


Figure 5.9 – Fonction de répartition du degré de satisfaction de l'orchestration

de probabilités (gaussienne dans notre exemple) de l'opération de service en question. Dans le but de simuler une dégradation relative au temps de réponse, nous multiplions par un facteur de cinq, l'écart-type (σ) de la distribution de probabilités (gaussienne) ; ceci ne concerne que la partie exécutée de l'orchestration. Ainsi, la génération aléatoire de valeurs à partir de ces distributions de probabilités simule le caractère non déterministe du temps de réponse. Nous aurons ainsi la possibilité d'avoir des bons comme des mauvais temps de réponse et ceci d'une manière complètement aléatoire. En ce qui concerne la fiabilité et la disponibilité, nous allons considérer une dégradation pour chacun de ces attributs qualité afin d'illustrer notre approche. Pour cela, nous allons supposer que l'opération de service S_2 est indisponible lors de la première invocation mais devient disponible à la deuxième tentative d'invocation (rappelons que nous avons considéré une stratégie de tolérance aux fautes pour ce service, voir section 5.4.2). Cette situation simule le cas réel lorsqu'il y a par exemple des problèmes de réseau lors de la première invocation de l'opération de service ou encore lorsque le service est très sollicité (il y a plusieurs requêtes) et que la requête en cours ne peut pas être servie. Nous avons donc, d'après les règles d'évaluation des stratégies de tolérance aux fautes que nous avons présentées dans la section 4.5.3, une disponibilité à l'exécution de l'opération de service S_2 égale à :

$$q_{av}(S_2) = 1 - \frac{t_0(S_2)}{t_0(S_2) + q_{rt}(S_2)} \ .$$

Sachant que la valeur aléatoire générée pour le temps de réponse de S_2 (représentant la mesure) est $q_{rt}(S_2) = 1.11$ et en prenant $t_0(S_2) = 1.5 \times \mu_{rt}(S_2)$, nous obtenons une valeur de la disponibilité de S_2 à l'exécution, égale à 0.27.

De la même manière pour l'opération de service S_5, nous supposons que la réponse de la première invocation n'est pas fiable tandis qu'à la suite de la deuxième tentative d'invocation, l'opération de service répond correctement. Rappelons que l'opération de

service S_5 est impliquée dans un patron de boucle (PC2) (voir figure 5.5). Dans le but d'illustrer l'application de la règle relative à l'évaluation de la stratégie de tolérance aux fautes, nous supposons que l'échec de l'opération de service S_5 survient à la deuxième itération du patron boucle. Notons que nous avons considéré le nombre d'itérations de la boucle, représentant le nombre d'opérations de production que contient l'affaire à produire (voir section 5.3), comme une variable aléatoire dont le domaine est $Dom(L) = \{1, 2, 3\}$. Nous prenons par la suite le cas d'une affaire qui contient deux opérations de production. Par conséquent, le nombre d'itérations de la boucle est déterministe à l'exécution et est égal à deux ($l = 2$). Ainsi, après l'exécution du patron boucle (PC2), nous avons, pour l'opération de service S_5, une première invocation réalisée avec succès (correspondant à la première itération du patron boucle), une deuxième invocation en échec (correspondant à la première invocation dans la stratégie de tolérance aux fautes, effectuée lors de la deuxième itération du patron boucle) et une troisième invocation (correspondant à la deuxième invocation dans la stratégie de tolérance aux fautes). Par conséquent, la fiabilité de l'opération de service S_5 dans la première itération du patron boucle est égale à 1 alors que dans la deuxième itération du patron boucle, elle est :

$$q_{rel}(S_5) = \frac{n_S}{n_T} = \frac{2}{3}$$

Le contexte d'application de notre approche étant défini, nous procédons dans la suite à l'évaluation de la qualité de l'orchestration au cours de son exécution.

5.5.2 Application de l'approche d'agrégation au cours de l'exécution

Nous avons pris dans la section précédente deux exemples représentant des dégradations relatives à la disponibilité et à la fiabilité se rapportant respectivement aux opérations de service S_2 et S_5. Nous présumons que les valeurs de la fiabilité et de la disponibilité des autres opérations de service exécutées sont égales à 1 et nous effectuons l'évaluation de la qualité de l'orchestration. Pour cela, nous situons l'instant « t » de l'évaluation après l'exécution de l'opération de service S_9 (voir figure 5.10). Il convient maintenant d'appliquer la méthode d'agrégation par les règles de patrons de *workflow* au cours de l'exécution (voir section 3.5). La situation de cet exemple, correspond à la deuxième situation que nous avons distinguée dans la section 3.5.2 (voir figure 3.12b, page 84). Ainsi, nous pouvons subdiviser l'orchestration en trois parties comme l'illustre la figure 5.10. Notons que la fiabilité et la disponibilité (qui prennent toujours des valeurs déterministes) sont agrégées par la méthode considérant les valeurs déterministes (voir section 3.3). Le temps de réponse global de l'orchestration, représenté par une variable aléatoire X_{rt}, est obtenu en agrégeant d'abord les temps de réponse des opérations de service impliquées dans chacune des trois parties prises séparément, puis les temps de réponse de ces trois parties. L'agrégation de la première partie (respectivement de la troisième partie) se fait en considérant les valeurs déterministes (voir section 3.3) (respectivement les variables aléatoires, voir section 3.4) des temps de réponse relatifs aux opérations de service impliquées dans ces

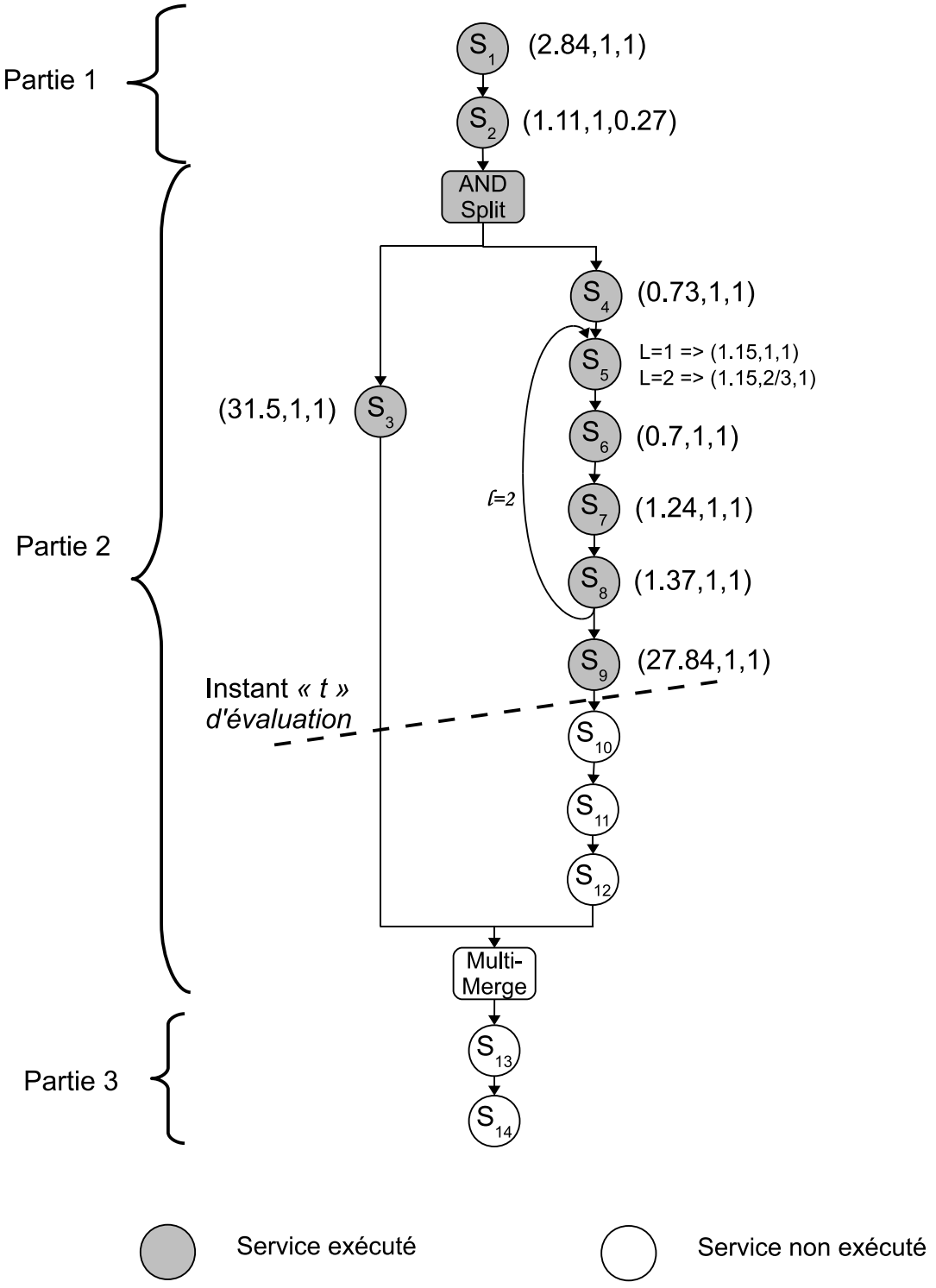


Figure 5.10 – Évaluation de la qualité de l'orchestration au cours de l'exécution

parties. Quant à la deuxième partie (décrite par le patron de composition PC5), nous appliquons le traitement que nous avons présenté pour les patrons de parallélisme (voir section 3.5.2).

Les vecteurs de valeurs d'attributs qualité relatifs aux opérations de service de la partie exécutée, sont représentées dans la figure 5.10. Ces vecteurs $(q_{rt}, q_{rel}, q_{av})$ dénotent respectivement les valeurs (mesures) du temps de réponse (généré de manière aléatoire), de la fiabilité et de la disponibilité. Les données de qualité des opérations de service qui ne sont pas encore exécutées, gardent leurs valeurs « théoriques » données dans le tableau 5.2. Au final, le résultat de l'agrégation par les règles de patrons de *workflow* est le suivant :

- temps de réponse global $\mu_{rt}(orch) = 75.91\ min$, $\sigma_{rt}(orch) = 2.22\ min$;
- fiabilité globale $q_{rel}(orch) = 0.39$;
- disponibilité globale $q_{av}(orch) = 0.21$.

Nous constatons d'après ce résultat une dégradation affectant chacun des trois attributs qualité par rapport à leur valeur pendant la phase de conception (à savoir 70.37, 0.4, 0.54, voir section 5.4.2). Cependant, les valeurs de la fiabilité et de la disponibilité ne violent pas les valeurs minimum acceptables (égales à 0.2, voir $QoSL^{neutral}$ dans le tableau 5.3), alors que le temps de réponse, en moyenne, viole ces exigences ($q_{rt}^{neutral} = 70$). Voyons maintenant la répercussion de cette dégradation sur le degré de satisfaction. En prenant en compte le modèle de préférences construit dans la section 5.4.1, la détermination de la distribution de probabilités du degré de satisfaction donne le résultat suivant :

$$\mu_{DS}(orch) = 0.01 \qquad\qquad \sigma_{DS}(orch) = 0.02$$

L'interprétation qui en découle est que le degré de satisfaction, en moyenne, présente une satisfaction des exigences du client ($\mu_{DS}(orch) > 0$), bien que la moyenne soit extrêmement faible voire nulle. Cela est justifié car le temps de réponse possède un poids d'importance relativement faible par rapport à la fiabilité et à la disponibilité (voir équation 5.9). Ainsi, si nous avions adopté la condition 4.28 (page 119) de déclenchement de l'adaptation dynamique, le système de supervision n'aurait pas décidé d'adapter l'orchestration.

Ces données qualitatives sont présentées à titre d'information. En réalité, le système de supervision ne surveille que l'état de la condition de déclenchement (vérifiée par le déclencheur, voir figure 3.1). Pour cela, il convient d'appliquer la stratégie de surveillance adoptée et de réaliser les actions nécessaires si besoin. Avant cela, nous revenons sur l'impact de la représentation de l'attribut qualité du temps de réponse. La valeur du temps de réponse calculée en considérant les mesures de la partie exécutée et uniquement les moyennes de la partie non exécutée, est égale à 69.44 *min* (contrairement à la moyenne obtenue précédemment qui est égale à 75.91 *min*). Ceci montre qu'aucun des attributs qualité dans ce cas ne viole les contraintes globales, et donc le système de supervision n'aurait pas décidé d'adapter l'orchestration.

5.5.3 Mise en œuvre de la gestion de l'adaptation dynamique

Rappelons que la stratégie de surveillance choisie pour cet exemple d'illustration, est la stratégie de surveillance réactive avec un seuil de probabilité égale à 20%. Après avoir agrégé les mesures des attributs qualité (bloc agrégation dans figure 4.4, page 114) et obtenu la distribution de probabilités du degré de satisfaction à l'instant « t » de l'évaluation, le déclencheur (voir figure 4.4) se charge de vérifier la condition de déclenchement donnée par l'expression 4.26. Pour cet exemple, la vérification de cette condition donne :

$$P(DS < 0) = 0.24\,,\ \text{donc}\ P(DS < 0) > s$$

Ainsi, le déclencheur signale qu'une intervention doit être menée afin de remédier aux dégradations survenues et assurer la préservation de la satisfaction des exigences au long de l'exécution. Notons que nous pouvons démontrer par les calculs faits (non reproduits ici) que suivant les données de simulation utilisées, ce n'est qu'à cet instant « t » de l'exécution que le déclencheur réclame pour la première fois une intervention. L'intervention se traduit par l'adaptation dynamique de l'orchestration en intégrant un ou plusieurs scénarios d'évolution que nous avons exposés dans la section 4.5.3. Le choix des scénarios d'évolution adéquats à mettre en place, est réalisé par la fonction de décision (voir figure 4.4). Dans notre exemple illustratif, nous mettons en place deux scénarios d'évolution. Le premier concerne les opérations de service S_{10} et S_{11} et a pour objectif d'améliorer le temps de réponse global de l'orchestration. Cela consiste à remplacer les opérations de service en question par d'autres opérations de service équivalentes ayant une distribution de probabilités du temps de réponse « meilleure » (par exemple une moyenne et un écart-type plus faibles). Concrètement, nous allons changer, pour la simulation, la moyenne et l'écart type de la distribution de probabilités de S_{10} (respectivement de S_{11}) par $\mu_{rt}(S_{10}) = 3\ min$ et $\sigma_{rt}(S_{10}) = 0.3\ min$ (respectivement $\mu_{rt}(S_{11}) = 15\ min$ et $\sigma_{rt}(S_{11}) = 1.5\ min$). Le deuxième scénario d'évolution concerne les opérations de service S_{13} et S_{14}. Il s'agit de mettre en place une stratégie de tolérance aux fautes d'invocation en boucle, avec un nombre de tentatives d'invocation égal à deux, et ce, dans le but d'améliorer la fiabilité et la disponibilité des opérations de service en question.

Dans le but d'analyser l'impact de l'adaptation réalisée, nous effectuons de nouveau l'évaluation de la qualité globale de l'orchestration au même instant « t ». Le résultat est le suivant :

- temps de réponse global : $\mu_{rt}(orch) = 69.38\ min$, $\sigma_{rt}(orch) = 1.49\ min$;
- fiabilité globale : $q_{rel}(orch) = 0.54$;
- disponibilité globale : $q_{av}(orch) = 0.23$;
- degré de satisfaction : $\mu_{DS}(orch) = 0.10$, $\sigma_{DS}(orch) = 0.01$;
- vérification de la condition de déclenchement : $P(DS < 0) = 0$.

En conclusion, l'approche de supervision que nous avons proposée, permet de préserver et d'assurer la satisfaction des exigences du client, et ce, tout au long de l'exécution

des orchestrations de services. Nous avons utilisé pour la simulation des valeurs fictives d'attributs qualité. Les algorithmes relatifs à la discrétisation, transformation inverse, réduction du domaine, opérations d'agrégation, pour les variables aléatoires ainsi que ceux relatifs à la détermination du modèle de préférences, ont été développés sous MATLAB. Dans ce sens, l'objectif est d'exposer ces algorithmes sous forme d'opérations de service et de services pour être invoqués à la demande.

5.6 Synthèse

Nous avons présenté, dans ce chapitre, la mise en œuvre d'une architecture orientée services dans un domaine particulier, le pilotage d'ateliers de production (domaine du MES, section 5.2). Notre contribution dans ce cadre, était la définition d'une démarche IDM (section 5.2.2) pour la solution logicielle MES comportant trois niveaux, à savoir le niveau métier, le niveau fonctionnel et le niveau technologique. Nos travaux ont concerné dans ce contexte, les niveaux fonctionnel et technologique. Chacun de ces deux niveaux est conçu selon une architecture orientée service. Pour atteindre cet objectif, nous avons établi une cartographie de services MES (section 5.2.3) qui couvre l'ensemble des fonctionnalités métiers offertes par les partenaires (éditeurs de progiciels) du projet MES. Nous avons regroupées ces fonctionnalités sous forme de 23 services possédant un total de 242 opérations, conformes au standard du Web 2.0 et à la norme ISA-95 relative au domaine du MES. Les services et les opérations de service sont répartis sur quatre domaines métiers et un cinquième domaine transversal. Cela a permis d'avoir un catalogue de fonctionnalités MES (opérations de service) parmi lesquelles, les clients pourront choisir et orchestrer les opérations de service qui les intéressent, selon les besoins métiers (à la demande). L'implémentation d'une partie de ces travaux a conduit au produit MESTRIA, une solution MES flexible (au sens où sa couverture fonctionnelle varie selon les besoins requis), conforme aux standards du Web 2.0 et ISA-95. Cette solution est représentée par un ensemble de scénarios métiers réalisés par des orchestrations de services dont un exemple simplifié a été présenté dans la section 5.3.

Nous avons ensuite appliqué notre approche de supervision sur cet exemple d'orchestration de services (section 5.4). Nous avons exposé dans un premier temps, les étapes de configuration nécessaires à la mise en œuvre de l'approche de supervision. Cela consiste principalement en la définition des exigences du client, la construction du modèle de préférences (sur lequel repose la supervision) et le choix de la stratégie de surveillance (réactive ou préventive). Nous avons ainsi détaillé la construction du modèle de préférences (section 5.4.1) qui se base uniquement sur les informations (préférences et intensités de préférences) recueillies du client. Nous avons également comparé notre modèle de préférences avec les travaux existants dans la littérature [Fakhfakh et al., 2012]. Nous avons ainsi montré, que notre modèle de préférence se distingue par la prise en compte à la fois, des préférences du client (grâce à la méthode MACBETH) dans la normalisation (construction des fonctions d'utilité), et des dépendances préférentielles (grâce à l'exploitation de

l'intégrale de Choquet 2-additive). Les tracés donnés en figures 5.6 et 5.7 illustrent ces résultats.

Nous avons appliqué dans un second temps, l'approche d'agrégation globale que nous avons proposée, pendant la phase de conception (section 5.4.2). Dans ce contexte, nous avons montré l'intérêt de représenter le temps de réponse par une variable aléatoire (pour les opérations de service non exécutées). Soulignons ici, que notre approche ne pose aucune restriction ni sur la nature des attributs qualité (déterministe ou non), ni sur le type des distributions de probabilités dans le cas où les attributs qualité sont non déterministes. L'application de l'approche d'agrégation pendant la phase de conception présente également d'autres intérêts (section 5.4.2). Ces derniers résident par exemple dans la validation de la conception de l'orchestration en terme de qualité ou encore, dans la comparaison de plusieurs orchestrations correspondant au même processus métier [Fakhfakh et al., 2011b], etc.

Par ailleurs, le cadre général (voir figure 3.1, page 52) dans lequel s'inscrit notre contribution, est la supervision des orchestrations de services tout au long de leur exécution (section 5.5). Nous avons ainsi mis en œuvre notre approche de supervision à travers un exemple d'exécution (section 5.5.1). Pour cela, nous avons simulé des dégradations relatives à chacun des trois attributs qualité. Nous avons ensuite illustré l'évaluation de la qualité globale de l'orchestration à un instant t de l'exécution (section 5.5.2). Enfin, nous avons appliqué une des deux stratégies de surveillance que nous avons proposées, et montré que l'on peut rétablir le degré de satisfaction grâce aux scénarios d'évolution définis. Ceci dans le but d'éviter qu'il y ait une éventuelle insatisfaction des exigences du client à la fin de l'exécution. Notons qu'aucun des travaux existants dans la littérature n'a pris en compte les préférences du client (satisfaction) dans la surveillance de la qualité des orchestrations de services comme le recommande la norme ISO 9001:2008 [ISO/TC 176, 2008].

Chapitre 6

Conclusions et perspectives

6.1 Conclusions

Les travaux de cette thèse portent sur une approche de supervision de la qualité des orchestrations de services. Dans ce contexte, nos travaux ont ciblé trois problématiques différentes (section 2.4) :

1. l'évaluation des attributs qualité portant sur l'orchestration de globale. Nous visons plus précisément ici les approches basées sur les règles de patrons de *workflow* [C. Jaeger, 2007, Cardoso et al., 2002, Rosenberg, 2009, Coppolino et al., 2007, Hwang et al., 2007] ;
2. l'interprétation de la qualité globale de l'orchestration de services ;
3. la surveillance et la gestion de la qualité des orchestrations de services au cours de l'exécution.

Nous avons vu que les travaux liés à l'agrégation des attributs qualité par les règles de patrons de *workflow*, présentent quelques limitations (sections 2.4.3 et 2.4.4). Ces limitations concernent d'une part le nombre de patrons de composition pris en compte dans ces travaux, et d'autre part la représentation des attributs qualité, notamment la représentation du temps de réponse au cours de l'exécution [Canfora et al., 2008, Szydlo and Zielinski, 2008, Qiao et al., 2009]. Nous avons remédié à ces limitations en proposant onze patrons de composition (section 3.3.2), exprimés à l'aide des quarante trois patrons de *workflow* [Russell et al., 2006a] résultant de la révision des vingt patrons de *workflow* initiaux [van der Aalst et al., 2003]. Nous avons également proposé treize nouvelles règles d'agrégation (voir tableaux 3.3 et 3.4) pour les attributs qualité que nous avons considérés, dont huit de ces règles d'agrégation n'étaient pas définies pour certains patrons de composition (voir tableau 3.2). Nous avons opté ensuite pour une représentation probabiliste des attributs qualité non déterministes, comme le temps de réponse, en particulier durant l'exécution. En prenant l'exemple du temps de réponse, nous avons montré que cette représentation permet d'éviter des erreurs non négligeables dans l'agrégation pendant la phase de conception (section 5.4.2) et au cours de l'exécution (section 5.5.2).

Tableau 6.1 – Bilan des règles d'agrégation de la littérature pour le temps de réponse, la fiabilité et la disponibilité

Critères	**Travaux proposant des règles d'agrégation**					**Travaux exploitant des règles d'agrégation**			**Nos travaux**
	[Cardoso et al., 2002]	[C. Jaeger, 2007]	[Coppolino et al., 2007]	[Hwang et al., 2007]	[Rosenberg, 2009]	[Canfora et al., 2008]	[Szydlo and Zielinski, 2008]	[Qiao et al., 2009]	
Patrons de composition									
PC1- Séquence	√	√	√	√	√	√	√	√	√
PC2- Boucle	√	√	√	√	√	√			√
PC3- XOR-XOR	√	√	√	√	√	√			√
PC4- AND-AND	√	√	√	√	√	√			√
PC5- AND-AND			√						√
PC6- AND-m/n	√	√	√						√
PC7- AND-XOR		√	√	√					√
PC8- OR-OR		√	√						√
PC9- OR-OR			√						√
PC10- OR-m/n		√	√						√
PC11- OR-XOR		√							√
Attributs qualité									
temps de réponse	(moyennes)	(min, max)		(probabiliste)	(moyennes)	(moyennes)	(moyennes)	(moyennes)	(probabiliste)
fiabilité	√		√	√		√			√
disponibilité		√			√	√	√		√
Période du cycle de vie de l'orchestration									
conception	√	√	√	√	√	√	√	√	√
au cours d'exécution						√	√	√	√

Notons que la fiabilité et la disponibilité sont des cas particuliers du fait que leur valeur elle-même représente une probabilité. Elles ne peuvent donc pas être représentées par des distributions de probabilités sauf si l'on considère qu'elles dépendent du temps, ce qui n'est pas communément considéré dans la littérature relative aux AOS (voir section 2.3.1). La sélection des trois attributs qualité que nous avons considérés, regroupe à la fois des grandeurs déterministes (fiabilité et disponibilité) et des grandeurs probabilistes (temps de réponse). Ceci montre que le traitement que nous avons présenté pour les différentes phases du cycle de vie de l'orchestration, à savoir pendant la conception (section 3.4), durant l'exécution (section 3.5) ou à la fin de l'exécution (section 3.3), peut s'appliquer à n'importe quel autre attribut qualité (qu'il soit déterministe ou non). Le tableau 6.1 présente un bilan de nos apports par rapport aux travaux relatifs à la première problématique. Le symbole « $\surd$ » mentionne que le critère en question est pris en compte dans les travaux correspondants. En ce qui concerne le temps de réponse, nous avons indiqué directement la manière dont il est représenté (soit par des moyennes statistiques, soit par des distributions de probabilités) lorsqu'il est pris en compte. Les cellules du tableau sont laissées vides lorsque le critère n'est pas pris en compte par les travaux correspondants. Enfin, nous pouvons constater que nos travaux améliorent en partie chacune des contributions existantes (voir tableau 6.1).

Par ailleurs, les travaux proposant des règles d'agrégation de patrons de *workflow* exposent une difficulté d'interprétation de la qualité globale de l'orchestration. Pour remédier à cette limitation, nous avons proposé d'agréger les valeurs des différents attributs qualité, sous la forme d'un degré de satisfaction (sections 4.3 et 4.4). Bien que certains travaux [Taher et al., 2005b,Herssens et al., 2008,Canfora et al., 2008,Szydlo and Zielinski, 2008] aient proposé, dans un contexte de sélection des services, des méthodes pour agréger les valeurs de différents attributs qualité, chacune de ces méthodes présente certaines limitations. Ces limitations concernent notamment la manière avec laquelle on construit les fonctions d'utilité marginales dédiées à la normalisation des valeurs d'attributs qualité, et l'opérateur d'agrégation considéré. La plupart de ces méthodes se basent sur des fonctions d'utilité linéaires [Taher et al., 2005b,Herssens et al., 2008]. Quant à la construction de la fonction d'utilité globale (détermination des paramètres de l'opérateur d'agrégation), elle est souvent laissée à la charge de l'utilisateur [Taher et al., 2005b, Canfora et al., 2008]. Nous avons effectué une comparaison de notre modèle de préférences avec ces travaux (section 5.4.1), et nous avons montré que l'on peut avoir, dans certains cas, des erreurs de sur-estimation ou de sous-estimation de la satisfaction du client par rapport à notre modèle de préférences. Nous estimons que, grâce *(i)* au recours à l'expression des préférences dans la construction des fonctions d'utilité (et la construction de la fonction d'agrégation) et *(ii)* à la prise en compte des dépendances préférentielles à l'aide de l'opérateur de l'intégrale de Choquet 2-additive, notre modèle de préférences offre une meilleure représentation de la satisfaction réelle du client. Le tableau 6.2 présente un bilan de nos contributions par rapport aux travaux liés à ce contexte. Le « - » signifie que le critère en question n'a pas

Tableau 6.2 – Bilan des travaux portant sur l'agrégation des valeurs de différents attributs qualité

Critères	[Taher et al., 2005b]	[Canfora et al., 2008]	[Szydlo and Zielinski, 2008]	[Herssens et al., 2008]	Nos travaux
Fonctions d'utilité marginales basées sur l'expression des préférences	-	-	?	-	+
Fonction d'utilité globale basée sur l'expression des préférences	-	-	+	+	+
Prise en compte des dépendances préférentielles	-	-	-	+	+

« - » : non pris en compte, « + » : pris en compte, « ? » : non mentionné dans les travaux.

été étudié par les travaux correspondants, tandis que « + » signifie que le critère a été pris en compte. Pour certains travaux, la prise en compte ou non de certains critères n'est pas explicitée ; cela est noté « ? ». Au final, nous pouvons conclure que notre modèle de préférence montre l'amélioration qu'apporte notre modèle de préférences par rapport aux travaux existants (voir tableau 6.2).

En ce qui concerne la dernière problématique, la gestion de la qualité des orchestrations au cours de l'exécution (section 2.4.5), aucun des travaux actuels ne prend en compte les préférences du client dans la surveillance de la qualité des orchestrations. Notre approche qui se base sur la distribution de probabilités du degré de satisfaction pour surveiller la qualité de l'orchestration au cours de l'exécution, est une première étape vers un système de supervision orienté client. Nous avons montré à travers l'exemple traité, qu'il est plus judicieux de se baser sur la distribution de probabilités du degré de satisfaction (de l'orchestration au regard des exigences du client) que de se contenter de la vérification de la satisfaction des contraintes (globales) liées à chacun des attributs qualité (voir section 5.5). De cette manière, le système de supervision basé sur le modèle de préférences établi, remplace le client dans le sens où les décisions/actions qu'il effectue sont conformes aux attentes du client. Ceci bien entendu, dans le but d'assurer la satisfaction du client (ses exigences en termes de qualité), qui représente une des préoccupations et un des challenges de toutes les entreprises aujourd'hui, face à la forte concurrence du marché.

Enfin, la plupart des travaux, comme par exemple [Canfora et al., 2008], proposent la substitution des opérations de service en cas d'indisponibilité à l'exécution. L'indisponibilité des opérations de service est souvent due à des problèmes de réseaux ou à une surcharge de requêtes du côté fournisseur. Pour cette raison, nous pensons que si l'on invoque de nouveau le service, il est possible que l'opération de service souhaitée soit disponible. Ceci

permet ainsi d'éviter les coûts liés à la recherche de services équivalents, à l'établissement de nouveaux contrats (SLA), etc. Dans ce but, nous avons étendu à la disponibilité les stratégies de tolérance aux fautes définies pour la fiabilité, et nous avons également étendu les règles d'agrégation permettant de les estimer aussi bien pendant la conception qu'au cours de l'exécution.

En dernière analyse, l'approche de supervision que nous avons proposée dans cette thèse est générique. Elle est indépendante de tout langage d'orchestration dans la mesure où elle s'appuie sur les patrons de *workflow* génériques pouvant être implémentés ou non par tout langage d'orchestration (voir section 2.2.2). De surcroît, elle ne pose aucune restriction, ni sur les attributs qualité à prendre en compte, ni dans la manière dont on les représente. En effet, n'importe quel attribut qualité peut être pris en compte (contrairement aux approches basées sur des modèles probabilistes, voir section 2.4.2) et peut être représenté par n'importe quelle distribution de probabilités (ce qui n'est pas le cas par exemple dans [Menascé et al., 2010]).

Cependant, notre approche ainsi que toutes les approches basées sur les règles d'agrégation de patrons de *workflow*, sont destinées uniquement aux orchestrations structurées de services (voir section 3.3.4), ce qui limite leur champs d'application. D'autre part, bien que notre modèle de préférences se distingue par sa précision par rapport aux travaux existants, il nécessite l'implication du client pour l'expression de ses préférences et de ses intensités de préférences. Cela peut être vu comme un point faible, notamment lorsque l'on considère plusieurs attributs qualité ce qui implique donc davantage de paramètres à déterminer et par suite plus d'informations à obtenir du client. Toutefois, le recueil des préférences du client est réalisé une seule fois, au moment de la configuration du système de supervision. Par conséquent, le système de supervision est complètement autonome par la suite. En outre, le modèle de préférences établi peut également servir pour d'autres orchestrations supportant le même processus métier [Fakhfakh et al., 2011b]. En effet, notre approche de supervision permet également de comparer plusieurs modèles d'orchestration pendant la phase de conception, en évaluant la condition de déclenchement de l'adaptation de la stratégie de surveillance choisie (par exemple, $e = P_{DS}(DS < 0)$). Dans ce cas, le modèle d'orchestration qui a la valeur de e la plus faible, est le meilleur. Enfin, les approches de sélection des services se basent de plus en plus aujourd'hui sur les préférences des clients [Sathya et al., 2011, Zhenehen et al., 2011] et nécessitent donc entre autres l'expression des préférences du client. Ainsi, nous considérons que cette implication du client ne représente pas un handicap majeur pour la mise en œuvre de notre approche.

Par ailleurs, nous avons identifié un phénomène où l'application de nos stratégies de supervision (c'est-à-dire la surveillance par rapport à la distribution de probabilités du degré de satisfaction) peut mener à un échec dans le sens où certaines dégradations, en particulier relatives aux temps de réponse des opérations de services, peuvent être masquées. Ce phénomène apparaît notamment lorsque les ordres de grandeur des temps de

réponse des différentes opérations de service impliquées dans l'orchestration, sont significativement différents. À titre d'exemple, revenons au processus MES (section 5.3) et considérons l'intégration d'un processus d'apprivoisement de matières premières après la création de l'affaire en production. La durée de ce processus peut être de l'ordre de la journée, voire de la semaine, alors que la durée du reste des opérations de service est de l'ordre de la minute. Ceci implique que la durée du processus global est de l'ordre de la journée ou de la semaine. Par conséquent, les dégradations du temps de réponse dues aux autres opérations de service (qui ont des temps de réponse faibles), bien qu'elles puissent être relativement importantes par rapport à leur durée, sont négligeables. Ceci ne permet donc pas de détecter la dégradation de la qualité de ces opérations de service. Autrement dit, le degré de satisfaction est moins sensible à la variation des temps de réponse faibles en présence de temps réponse importants. Ce problème est aussi vrai pour d'autres travaux [Ben Halima et al., 2008, Qiao et al., 2009].

En revanche, certaines solutions peuvent être envisagées pour les limitations précédemment identifiées, c'est ce que nous proposons comme perspectives de cette thèse.

6.2 Perspectives

Plusieurs perspectives peuvent être dégagées au terme de ces travaux. Certaines concernent l'expérimentation et l'outillage de notre approche de supervision, d'autres ciblent l'amélioration et l'extension de notre proposition.

6.2.1 Expérimentation et outillage

Le premier travail à conduire à l'issue des travaux de cette thèse, consiste en l'expérimentation de notre approche de supervision sur d'autres cas, scénarios avec des données réelles des attributs qualité. Une des expériences à mener, consiste à faire varier les paramètres s (seuil de probabilité) et/ou β (marge de prévention pour la stratégie de supervision préventive) dans le but de déterminer la bonne configuration permettant de superviser l'orchestration en question tout en ayant le moins de violations des exigences du client. Pour chaque valeur de s et/ou de β, on exécute l'orchestration x fois. À la fin de chaque exécution de l'orchestration, on calcule le degré de satisfaction correspondant. Si le degré de satisfaction est positif, cela veut dire que les exigences du client sont respectées, sinon elles sont violées. La bonne configuration de s et/ou β est celle qui donne le moins de violations enregistrées sur x exécutions de l'orchestration. Le nombre de déclenchements de l'adaptation dynamique durant chaque exécution de l'orchestration, peut être pris en compte pour déterminer la bonne configuration du système de supervision : moins on déclenche l'adaptation dynamique durant l'exécution (tout en assurant la satisfaction des exigences du client bien entendu), meilleure est la configuration.

D'autre part, l'évaluation et la surveillance de la qualité de l'orchestration au cours de l'exécution dans notre approche, se fait après l'exécution de chaque opération de service impliquée dans l'orchestration. Ceci peut être inutile, notamment au début de l'exécution

de l'orchestration vu que l'on a peu de mesures et que la qualité de l'orchestration peut évoluer. Dans ce sens, nous pouvons envisager de définir des points de contrôle (*checkpoints*) [Leitner et al., 2009] précisant les instants où l'évaluation de la qualité doit s'effectuer. Dans ce cas, un compromis doit être considéré entre des points de contrôle au début de l'exécution et à la fin de l'exécution. D'une part, bien que la définition des points de contrôle au début de l'exécution offre plus de possibilités pour l'adaptation, l'estimation de la qualité attendue (à la fin de l'exécution de l'orchestration) est peu précise puisque nous avons peu de mesures. D'autre part, la définition des points de contrôle vers la fin de l'exécution laisse peu de choix pour réagir et donc peu de chances de rétablir la qualité de l'orchestration. Ainsi, des expériences à mener pourraient nous guider dans le choix des positions des points de contrôle.

Pour effectuer ces expériences, il faut d'abord considérer un langage d'orchestration et en développer un analyseur syntaxique (*parser*) permettant d'identifier les patrons de composition utilisés dans l'orchestration en question (voir section 3.3.4). Ensuite, il convient de développer les fonctions du système de supervision (voir figure 4.4, page 114) que nous n'avons pas pu développer par faute de temps, à savoir l'acquisition des mesures des attributs qualité ainsi que la décision suite au déclenchement de l'adaptation dynamique. Nous avons cité dans la section 3.2.2 quelques techniques permettant la mesure des attributs qualité que nous avons considérés. D'autres idées/pistes concernant la fonction de décision sont abordées dans la section suivante. Notons que les mécanismes d'adaptations dynamique des orchestrations ainsi que le langage d'orchestration, ont été le sujet de travaux précédents et actuels [Pourraz, 2007, Verjus et al., 2011] dont le résultat est un langage d'orchestration *PXL* permettant l'évolution dynamique des orchestrations de services en cours d'exécution (voir section 2.2.2). Nous envisageons également d'exposer les algorithmes relatifs à la discrétisation des variables aléatoires continues, la transformation inverse de variables discrétisées, la réduction de la taille du domaine des variables discrètes, les différentes opérations d'agrégation, la construction des fonctions d'utilité marginales et de la fonction d'utilité globale et la détermination de la distribution de probabilités du degré de satisfaction, qui sont aujourd'hui implémentés sous le logiciel MATLAB, en services Web [Mathworks, 2012] pour être invoqués à la demande par le système de supervision.

6.2.2 Fonction de décision du système de supervision

En ce qui concerne la fonction de décision du système de supervision, rappelons que son objectif est de choisir le scénario d'évolution adéquat correspondant à la dégradation de qualité survenue.

La question qui se pose dans un premier temps est quels attributs qualité faut-il cibler par les scénarios d'évolution ? La réponse à cette question permet par exemple de choisir la mise en place de stratégies de tolérances aux fautes relatives à la fiabilité et/ou à la disponibilité lorsque celles-ci sont à l'origine de la dégradation. Dans le cas d'une dégradation affectant le temps de réponse, la réponse à cette question permet par exemple d'opter, soit pour un ré-ordonnancement du reste de l'orchestration, soit pour un remplacement de

quelques opérations de service. Pour cela, si nous avions uniquement des attributs qualité déterministes (c'est-à-dire représentés par des valeurs fixes), nous pourrions nous inspirer des travaux de [Berrah et al., 2008] pour déterminer les attributs qualité concernés par le scénario d'évolution et de combien, en valeur, on doit améliorer la satisfaction de chacun de ces attributs de l'orchestration, plus précisément de la partie restante non exécutée, ceci dans le but de respecter les exigences du client ($DS > 0$) ou plus généralement la condition de déclenchement de l'adaptation. Le problème revient à minimiser la fonction objectif suivante :

$$\min(\delta_{rt} + \delta_{rel} + \delta_{av}) \tag{6.1}$$

où δ_i est la valeur de l'amélioration de la satisfaction de l'attribut qualité i. Des contraintes spécifiant la limite possible d'amélioration de chacun des attributs qualité peuvent être ajoutées afin d'avoir une solution réalisable. Ce problème d'optimisation peut se mettre sous une forme linéaire et être résolu par un algorithme de simplexe [Berrah et al., 2008]. Généralement, la solution correspond à un δ_i^* relatif à l'attribut qualité qui possède le poids d'importance le plus élevé. Nous avons présenté ici le cas des attributs qualité déterministes, en particulier nous avons considéré le temps de réponse comme un attribut déterministe (représenté par sa moyenne par exemple). Or, dans notre approche, nous avons considéré le temps de réponse comme une variable aléatoire représentée par sa distribution de probabilités. Ainsi, le challenge réside en l'extension de ce problème pour le cas des variables aléatoires. Il nous semble que le cadre d'optimisation stochastique [Kall and Wallace, 1994, Shapiro et al., 2009] permettrait d'aborder ce problème.

La problématique qui apparaît à la suite de l'identification de la cible du scénario d'évolution (c'est-à-dire l'identification des attributs qualité à améliorer et éventuellement de combien il faut les améliorer), consiste à choisir le scénario d'évolution (l'action de correction) à mettre en place parmi tous les scénarios possibles. Cette problématique revient à une problématique de choix (P.α), une des problématiques de référence dans d'aide à la décision (voir section 4.2.2). Outre les attributs qualité, d'autres critères devraient éventuellement être identifiés, comme par exemple l'usage des ressources, le coût de la mise en place d'un scénario, etc. Une étude approfondie de cette problématique de référence [Roy, 1985, Figueira et al., 2005] et des méthodes permettant de la résoudre peut être envisagée afin d'élaborer une procédure automatisée (à implémenter dans le système de supervision) du choix du scénario d'évolution.

Une autre piste en relation avec les méthodes d'aide à la décision multi-critères, concerne l'amélioration de l'interprétation de la qualité des orchestrations.

6.2.3 Amélioration de l'interprétation de la qualité

Nous avons vu que l'approche que nous avons présentée dans cette thèse permet de fournir une seule information de haut niveau, le degré de satisfaction. Ce degré de satisfaction

permet de faciliter l'interprétation et l'appréciation de la qualité globale de l'orchestration (l'ensemble des valeurs des différents attributs qualité) ; un exercice qui peut se montrer complexe du point de vue cognitif [Bouyssou et al., 2006], notamment en présence de multiples attributs qualité. En outre, cette information de haut niveau permet de situer la qualité (perçue) de l'orchestration au regard des exigences du client, en particulier elle permet de distinguer si l'on est dans la zone d'insatisfaction, de satisfaction ou de sur-satisfaction des exigences du client en termes de qualité (voir figure 6.1). La question qui peut se poser à titre d'exemple est : que signifie un degré de satisfaction égal à 0.2 ou égal à 0.7 ? L'idée consiste donc à aller au delà de ces trois zones (ou classes) et à les départager encore en des zones qui auront plus de signification du point de vue client. Nous pourrions imaginer (voir figure 6.1) de définir par exemple une zone de bonne satisfaction, une zone de moyenne satisfaction et une zone de faible satisfaction. Cela correspond typiquement à une problématique de tri (P.β) parmi les problématiques de référence dans l'aide à la décision (voir section 4.2.2) qui mériterait ainsi d'être étudiée. Cependant, le but est purement synoptique à notre sens ; cela n'impactera pas notre approche de supervision, en particulier les stratégies de surveillance que nous avons proposées. Autrement dit, le comportement du système de supervision reste neutre/indifférent au regard des sous-zones au sein des trois zones principales (insatisfaction, satisfaction, sur-satisfaction).

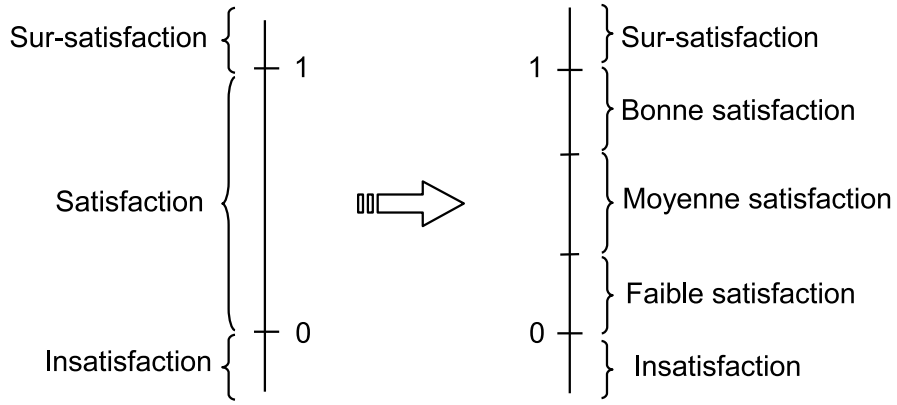


Figure 6.1 – Amélioration de l'interprétation du degré de satisfaction

La mise en œuvre éventuelle de la piste d'amélioration évoquée précédemment aurait lieu pendant la phase de conception, plus précisément lors de l'étape de configuration du système de supervision. Autrement dit, les classes (zones, catégories, etc) doivent être définies au préalable pendant cette étape de configuration avant la construction du modèle de préférences (fonctions d'utilité marginales et fonction d'utilité globale). Dans ce sens, nous pouvons dégager une autre piste qui concerne la configuration du système de supervision.

6.2.4 Configuration du système de supervision

La fonction d'utilité globale de notre modèle de préférences se base aujourd'hui sur l'opérateur de l'intégrale de Choquet 2-additive. Nous avons remarqué, lorsque nous avons

établi une comparaison de notre modèle de préférences au regard des travaux existants (section 5.4.1), que dans certains cas l'opérateur d'agrégation de la moyenne pondérée fournit presque les mêmes résultats que celui de l'intégrale de Choquet 2-additive (par exemple pour le $QoSL_4$ et $QoSL_7$, voir figure 5.7, page 150). Ceci nous amène à penser que pour certains clients, la moyenne pondérée suffirait éventuellement pour bien représenter leur satisfaction réelle (tout en gardant la même méthode pour la normalisation). Ainsi, nous pouvons envisager par exemple une sorte de questionnaire pour identifier si le client présente ou non des dépendances préférentielles vis-à-vis des attributs qualité. La question à se poser est typiquement : *est-ce que la satisfaction en regard d'un attribut qualité dépend des satisfactions d'autres attributs qualité ?* Cela nous permet, dans le cas d'une réponse négative, de réduire le nombre de paramètres à déterminer (pour construire la fonction d'utilité globale) et par conséquent, réduire l'implication du client. Dans ce sens, nous pourrions laisser l'opérateur d'agrégation configurable selon que le client présente ou non des dépendances préférentielles.

Par ailleurs, lors de la construction du modèle de préférences, nous avons supposé que les exigences du client en termes de qualité (les contraintes globales sur la qualité de l'orchestration) sont fixes et définies en amont pendant la phase de conception. Qu'en est-il lorsqu'elles sont variables et dans quelles situations se présente ce cas de figure ? C'est ce que nous abordons dans la section suivante.

6.2.5 Prise en compte des exigences variables

Nous avons identifié trois cas de figure où la prise en compte des exigences pourrait être utile. Nous commençons donc par présenter ces cas de figure avant de proposer une piste d'étude.

Dans cette thèse, le contrôleur du système de supervision (voir figure 4.4, page 114) a pour vocation de minimiser voire de compenser les effets des dégradations (externes) affectant la qualité de l'orchestration. Ceci est réalisé dans le cas où les exigences du client sont fixes (définies au préalable pendant la phase de conception). Ce problème s'identifie à la notion de *régulation* dans la théorie de contrôle des systèmes automatisés, où l'on suppose que la consigne (exigences du client) est fixe et que l'objectif du correcteur (système de commande) est de neutraliser les perturbations externes (non maîtrisables) affectant le système. Une autre fonction que peut réaliser un système de commande concerne l'*asservissement*, c'est-à-dire la poursuite par la sortie du système en question d'une consigne variable dans le temps [Kesraoui, 2010]. Dans ce sens, une extension de notre approche consistera à considérer des exigences variables du client. Cela permettra de couvrir plus de situations réelles, en particulier lorsque les besoins métiers du client changent au cours de l'exécution. En effet, l'évolution dynamique des orchestrations de services peut être réalisée à l'origine d'une dégradation de la qualité, sujet de nos travaux, comme elle peut être réalisée par décision directe du client pour étendre son processus mé-

tier et répondre à des nouveaux besoins métiers, etc. Dans ce cas, les exigences en termes de qualité vont potentiellement changer. La prise en compte des exigences variables permettrait ainsi de couvrir ce cas de figure.

Un autre cas de figure que nous pouvons soulever, s'adresse aux méthodes de sélection des services qui considèrent aujourd'hui que les attributs qualité des services sont des valeurs fixes ; qu'en est-il lorsqu'il s'agit de distributions de probabilités des attributs qualité ? Comment discriminer les services au regard des distributions de probabilités de leur attributs qualité ? Une solution qui peut être envisagée, consiste à déterminer la distribution de probabilités du degré de satisfaction de chacun des services candidats (en appliquant uniquement la deuxième phase de notre approche d'agrégation) et de comparer ensuite les services en analysant ces distribution de probabilités, par exemple en comparant les moyennes de degré de satisfaction de chacun des services ou en comparant les valeurs s_i de la probabilité d'avoir un degré de satisfaction supérieur à une certaine valeur ($s_i = P(DS > valeur)$), la contrainte locale. Or, afin de pouvoir effectuer cela pour tous les services impliqués dans l'orchestration, nous avons besoin potentiellement d'un modèle de préférences par service abstrait (fonctionnalité métier), notamment lorsque les exigences locales sur les fonctionnalités du processus sont différentes (exigences variables), ce qui est relativement fastidieux. Là encore, la prise en compte des exigences variables permettrait de surmonter cette difficulté.

Dans le même contexte, une solution similaire peut être envisagée pour résoudre le problème lié à la différence significative relative aux ordres de grandeur des temps de réponse des opérations de service impliquées dans l'orchestration (voir section 6.1). En effet, lorsque nous avons des sous-processus dans le processus global (comme l'exemple de processus d'approvisionnement évoqué précédemment) dont les ordres de grandeur de leurs attributs qualité sont très différents et donc des exigences de qualité différentes, nous pouvons considérer la supervision de ces sous-processus indépendamment, c'est-à-dire considérer chaque sous-processus comme une orchestration à part entière et appliquer l'approche de supervision. Cela implique qu'il faut établir un modèle de préférences par catégorie de sous-processus, c'est-à-dire l'ensemble des sous-processus ayant les mêmes exigences en termes de qualité. Là encore, nous aurons besoin de plusieurs modèles de préférences (comme dans le cas d'utilisation pour la sélection des service) puisque les exigences des sous-processus sont variables.

Comme nous pouvons le constater, la prise en compte des exigences variables pourrait apporter une solution à plusieurs problématiques de recherche que nous avons soulevées dans cette section. Si l'on présume que nous avons déjà élaboré un modèle de préférences relatif à un contexte métier donné, une piste de réflexion à explorer consiste à extrapoler d'autres modèles de préférences à partir de ce modèle existant. En effet, nous pensons que pour un contexte métier donné, l'importance des attributs qualité du point de vue

client, reste *a priori* la même. Ceci signifie que la deuxième partie du modèle de préférences (paramètres de l'opérateur d'agrégation) ne change pas. Par contre, c'est la satisfaction au regard des valeurs des attributs qualité (normalisation) qui peut varier d'un service à un autre, ou d'un processus à un autre puisque les contraintes de qualité changent. L'idée est donc de voir si nous pourrions extrapoler les fonctions d'utilité marginales de chacun des attributs qualité pour répondre aux nouvelles exigences. Une étape de validation et de test de la fidélité des modèles de préférences obtenus (c'est-à-dire vérification de la conformité des modèles de préférences avec la satisfaction réelle du client) devrait également être envisagée.

6.2.6 Étude des modèles d'orchestrations non structurés

Sous un autre aspect, comme nous l'avons précédemment mentionné (voir section 6.1), notre approche de supervision s'adresse aux orchestrations de services structurées. Bien que les modèles d'orchestration non structurés soient susceptibles d'avoir des situations de blocage à l'exécution (contrairement aux modèles structurés) [Kiepuszewski et al., 2000], cela n'empêche pas qu'ils doivent être supervisés. Ils méritent donc d'être étudiés et pris en compte. Nous avons vu qu'il est possible dans certains cas de transformer les modèles d'orchestrations non structurés en des modèles structurés [Kiepuszewski et al., 2000] (voir section 3.3.4). Ces transformations concernent généralement les boucles et les choix arbitraires et certains cas de parallélisme. Or, cela pourrait provoquer, à notre sens, un double compte de la qualité des services dupliqués (voir figure 3.8c, page 73). [Dumas et al., 2010] ont étudié deux types de non structuration dans les modèles d'orchestration, à savoir les boucles mono-entrée-multi-sorties (*Single-Entry-Multi-Exit Loop, SEMELoop*) et les graphes acycliques orientés (*Directed Acyclic Graph, DAG*). À titre d'exemple, la figure 3.8a (page 73) correspond à un DAG irréductible (c'est-à-dire qu'il n'admet pas de transformation structuré). Les auteurs de ces travaux ont proposé une méthode et des règles d'agrégation permettant d'évaluer les attributs qualité de ces composants non structurés. Par exemple, en ce qui concerne les composants DAG, la méthode consiste à trouver tous les chemins d'exécution possibles ainsi que la probabilité d'observer un chemin d'exécution donné. Ensuite, le composant DAG est traité comme un patron de choix, où chaque branche du patron correspond à un chemin d'exécution avec sa probabilité d'être exécuté. L'évaluation des attributs qualité pour ce composant correspond dans ce cas à une somme des valeurs des attributs qualité de chaque chemin d'exécution, pondérée avec les probabilités d'avoir ce chemin à l'exécution. Les attributs qualité considérés dans ces travaux correspondent au temps de réponse, au coût et à la fiabilité. Le coût (respectivement la fiabilité) de chaque chemin d'exécution est calculé comme étant la somme des coûts (respectivement le produit) de chaque service impliqué dans le chemin d'exécution. Quant au temps de réponse, il est calculé par la méthode du chemin critique [Zeng et al., 2004] (c'est-à-dire le chemin d'exécution qui a la plus longue durée). Ces travaux ainsi présentés peuvent constituer un point de départ pour la prise en compte des non structurations dans les modèles d'orchestration dans le but d'élargir le champ de couverture de

notre approche de supervision.

6.2.7 Vers un système de supervision métier

En dernier lieu, nous avons pris en compte, dans notre approche, des attributs qualité que nous avons recensés dans le domaine des architectures orientées services. Nous pensons que nous pourrons également prendre en compte des attributs qualité métiers (des indicateurs de performance métiers), comme par exemple le temps d'arrêt/panne machine, la quantité de produit en rebuts, etc, que l'on peut leur définir des règles d'agrégation relatifs aux patrons de composition et appliquer par la suite l'approche globale de supervision. On s'orientera ainsi vers un système de supervision métier (*Business Activity Monitoring*) [Lubinski, 2008, Oracle, 2009].

Publications

Conférences d'audience internationale avec actes

Fakhfakh, N., Verjus, H., and Pourraz, F. (2011). Qos aggregation in service orchestrations based on the choquet integral. In *Proceeding of the 8th IEEE International Conference on e-Business Engineering (ICEBE 2011).*, pages 77–84, Beijing, China.

Fakhfakh, N., Verjus, H., and Pourraz, F. (2011). Multi-criteria decision making method for quality of service aggregation. In *Proceedings of the Fifteenth IEEE International Entreprise Distributed Object Computing Conference (EDOC 2011), "The Enterprise Computing Conference"*, pages 203–212, Helsinki, Finland.

Fakhfakh, N., Pourraz, F., and Verjus, H. (2011). Quality of service aggregation in e-business applications. In *Proceedings of ICE-B 2011 International Conference on e-Business*, pages 100–110, Sevilla, Spain. SciTePress.

Fakhfakh, N., Verjus, H., Pourraz, F., and Moreaux, P. (2011). Measuring the satisfaction degree of quality attributes requirements for services orchestrations. In *The Fourth International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ 2011)*, pages 89–94, Budapest, Hungary.

Conférences d'audience nationale et francophone

Fakhfakh, N., Verjus, H., Pourraz, F., and Moreaux, P. (2010). Architectures orientées services pour les systèmes d'information d'entreprises agiles. In *Atelier SIRE (Systèmes d'Information des oRganisations Étendues) associé à la conférence INFORSID 2010*, Marseille, France.

Contribution à ouvrage

Fakhfakh, N., Pourraz, F., Verjus, H., and Moreaux, P. (2012). Client-oriented preferences model for QoS aggregation in service-based applications. In Obaidat, M. S. and Filipe, J., editors, *e-Business and Telecommucations*, Communications in Computer and Information Science, page 16. Springer-Verlag. (to appear).

Revue nationale

Verjus, H., Pourraz, F., and Fakhfakh, N. (2011). Cadre conceptuel pour la modélisation et la supervision d'architectures à base de services. Une approche pour l'ingénierie des SI à base de services. *Développement des SI à base de modèles: exigences, traçabilité et co-conception - Ingénierie des Systèmes d'Information - Revue des sciences et technologies de l'information*, 16(5):43–72.

Livrables de projet

Fakhfakh, N., Verjus, H., and Pourraz, F. (2011). Cartographie des services mes et de leurs opérations. Deliverable, MES Project Consortium, Annecy-le-Vieux, France.

Fakhfakh, N., Carnus, B., Verjus, H., and Pourraz, F. (2011). Description des services mes et de leurs opérations. Deliverable, MES Project Consortium, Annecy-le-Vieux, France.

Fakhfakh, N., Carnus, B., Verjus, H., and Pourraz, F. (2011). Exemples de mise en œuvre de la description des services mes et de leurs opérations. Deliverable, MES Project Consortium, Annecy-le-Vieux, France.

Pourraz, F., Verjus, H., and Fakhfakh, N. (2011). Illustration d'utilisation des services web de la cartographie et scénarios fonctionnels mes. Deliverable, MES Project Consortium, Annecy-le-Vieux, France.

Verjus, H., Pourraz, F., and Fakhfakh, N. (2011). Interopérabilité des progiciels du domaine du MES supportée par une approche à base de services - Avancement du projet au 31 décembre 2011. Technical report.

Verjus, H., Pourraz, F., and Fakhfakh, N. (2010). Interopérabilité des progiciels du domaine du MES supportée par une approche à base de services - Avancement du projet au 30 octobre 2010. Technical report.

Verjus, H., Pourraz, F., and Fakhfakh, N. (2009). Interopérabilité des progiciels du domaine du MES supportée par une approche à base de services - Avancement du projet au 30 septembre 2009. Technical report.

Bibliographie

[ISO, 1991] (1991). Quality management and quality assurance.

[Assaf Arkin, 2003] Assaf Arkin, I. (2003). Business process modeling language. http ://www.bpmn.org/Documents/BPML-2003.pdf.

[Bana e Costa et al., 2005] Bana e Costa, C., Corte, J., and Vansnick, J. (2005). On the mathematical foundation of MACBETH. In *Multiple Criteria Decision Analysis : State of the Art Surveys*, volume 78 of *International Series in Operations Research &amp ; Management Science*, pages 409–437. Springer New York.

[Bana e Costa and Vansnick, 1999] Bana e Costa, C. and Vansnick, J. (1999). The MACBETH approach : Basic ideas, software and an application. In Meskens, N. and Roubens, M., editors, *Advances in Decision Analysis*, pages 131–157. Kluwer Academic Publishers, Dordrecht.

[Ben Halima et al., 2008] Ben Halima, R., Guennoun, K., Drira, K., and Jmaiel, M. (2008). Providing predictive self-healing for web services : A qos monitoring and analysis-based approach. *Journal of Information Assurance and Security*, 3 :175–184.

[Berrah and Clivillé, 2007] Berrah, L. and Clivillé, V. (2007). Towards an aggregation performance measurement system model in a supply chain context. *Comput. Ind.*, 58(7) :709–719.

[Berrah et al., 2008] Berrah, L., Mauris, G., and Montmain, J. (2008). Monitoring the improvement of an overall industrial performance based on a choquet integral aggregation. *Omega*, 36(3) :340 – 351. Special Issue on Multiple Criteria Decision Making for Engineering.

[Booth et al., 2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). W3C working group note : Web services architecture. http ://www.w3.org/TR/ws-arch/.

[Borderie, 2006] Borderie, X. (2006). Expliquez-moi ... les propriétés ACID d'une base de données. http ://www.journaldunet.com/developpeur/tutoriel/theo/060615-theo-db-acid.shtml.

[Bouyssou et al., 2006] Bouyssou, D., Dubois, D., Pirlot, M., and Prade, H. (2006). *Concepts et méthodes pour l'aide à la décision, volume 3, analyse multicritère*. Hermès.

[B.V., 2012] B.V., N. B. (2012). International standard for the integration of entreprise and control systems.

[C. Jaeger, 2007] C. Jaeger, M. (2007). *Optimising Quality-of-Service for the Composition of Electronic Services*. PhD thesis, Berlin University, Germany.

[Canfora et al., 2005a] Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (2005a). An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1069–1075, New York, NY, USA. ACM.

[Canfora et al., 2005b] Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (2005b). Qos-aware replanning of composite web services. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '05, pages 121–129, Washington, DC, USA. IEEE Computer Society.

[Canfora et al., 2008] Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (2008). A framework for qos-aware binding and re-binding of composite web services. *J. Syst. Softw.*, 81 :1754–1769.

[Cardoso et al., 2002] Cardoso, J., Miller, J., Sheth, A., and Arnold, J. (2002). Modeling quality of service for workflows and web service processes. *Journal of Web Semantics*, 1 :281–308.

[Castellanos et al., 2003] Castellanos, M., Casati, F., Dayal, U., and Shan, M.-C. (2003). Intelligent management of slas for composite web services. In Bianchi-Berthouze, N., editor, *Databases in Networked Information Systems*, volume 2822 of *Lecture Notes in Computer Science*, pages 158–171. Springer Berlin / Heidelberg.

[Ciardo et al., 1993] Ciardo, G., Blakemore, A., Chimento, P. F., Muppala, J. K., and Trivedi, K. S. (1993). Automated generation and analysis of markov reward models using stochastic reward nets. *IMA Volumes in Mathematics and its Applications : Linear Algebra, Markov Chains, and Queueing Models / Meyer, C. ; Plemmons, R.J*, 48 :145–191.

[Cîmpan and Verjus, 2005] Cîmpan, S. and Verjus, H. (2005). Challenges in architecture centred software evolution. In *CHASE : Challenges in Software Evolution*, pages 1–4, Bern, Switzerland.

[Cîmpan et al., 2009] Cîmpan, S., Verjus, H., and Alloui, I. (2009). Approche centrée architecture pour l'évolution dynamique de systèmes d'information. *Technique et Science Informatiques - RSTI-TSI*, 28(5) :571–609.

[Clivillé, 2004] Clivillé, V. (2004). *Approche systèmatique et méthode multicritère pour la définition d'un système d'indicateurs de performance.* PhD thesis, Université de Savoie.

[Clivillé et al., 2007] Clivillé, V., Berrah, L., and Mauris, G. (2007). Quantitative expression and aggregation of performance measurements based on the macbeth multi-criteria method. *International Journal of Production Economics*, 105(1) :171–189.

[Collet, 2006] Collet, P. (2006). état de l'art sur la contractualisation et la composition. Livrable F-1.1, Projet RNTL FAROS.

[Committee, 2006] Committee, W. S. B. P. E. L. T. (April 2006). Wsbpel issue 42.

[Coppolino et al., 2007] Coppolino, L., Romano, L., Mazzocca, N., and Salvi, S. (2007). Web services workflow reliability estimation through reliability patterns. *Security and Privacy in Communications Networks and the Workshops,.*

[Cortellessa and Grassi, 2007] Cortellessa, V. and Grassi, V. (2007). Reliability modeling and analysis of service-oriented architectures. In *Test and Analysis of Web Services*, pages 339–362.

[Développement Construction, 2011] Développement Construction (2011). La mesure de la satisfaction-clients : un outil de pilotage marketing devenu indispensable.

[DISP, 2011] DISP, I. d. L. (2011). Mapping des entrées/sorties de la cartographie avec B2MML. Délivrable, MES Project Consortium, Villeurbanne, France.

[Dumas et al., 2010] Dumas, M., García-Bañuelos, L., Polyvyanyy, A., Yang, Y., and Zhang, L. (2010). Aggregate quality of service computation for composite services. In *ICSOC*, pages 213–227.

[Durate-Amaya, 2007] Durate-Amaya, H. (2007). *Tcows Canevas pour la composition de service web avec propriétés transactionnelles.* PhD thesis, Université Joseph Fourrier, Grenoble.

[Fakhfakh et al., 2011a] Fakhfakh, N., Carnus, B., Verjus, H., and Pourraz, F. (2011a). Description des services mes et de leurs opérations. Deliverable, MES Project Consortium, Annecy-le-Vieux, France.

[Fakhfakh et al., 2011b] Fakhfakh, N., Pourraz, F., and Verjus, H. (2011b). Quality of service aggregation in e-business applications. In *Proceedings of ICE-B 2011 International Conference on e-Business*, pages 100–110, Sevilla, Spain. SciTePress.

[Fakhfakh et al., 2012] Fakhfakh, N., Pourraz, F., Verjus, H., and Moreaux, P. (2012). Client-oriented preferences model for QoS aggregation in service-based applications. In Obaidat, M. S., Sevillano, J. L., and Filipe, J., editors, *e-Business and Telecommucations*, Communications in Computer and Information Science, page 16. Springer-Verlag. (to appear).

[Fakhfakh et al., 2011c] Fakhfakh, N., Verjus, H., and Pourraz, F. (2011c). Cartographie des services mes et de leurs opérations. Deliverable, MES Project Consortium, Annecy-le-Vieux, France.

[Fakhfakh et al., 2011d] Fakhfakh, N., Verjus, H., and Pourraz, F. (2011d). Multi-criteria decision making method for quality of service aggregation. In *Proceedings of the Fifteenth IEEE International Entreprise Distributed Object Computing Conference (EDOC 2011), "The Enterprise Computing Conference"*, pages 203–212, Helsinki, Finland.

[Fakhfakh et al., 2011e] Fakhfakh, N., Verjus, H., and Pourraz, F. (2011e). Qos aggregation in service orchestrations based on the choquet integral. In *Proceeding of the 8th IEEE International Conference on e-Business Engineering (ICEBE 2011).*, pages 77–84, Beijing, China.

[Fakhfakh et al., 2010] Fakhfakh, N., Verjus, H., Pourraz, F., and Moreaux, P. (2010). Architectures orientées services pour les systèmes d'information d'entreprises agiles. In *Actes de l'Atelier SIRE (Systèmes d'Information des oRganisations Étendues) associé à la conférence INFORSID 2010*, Marseille, France.

[Fakhfakh et al., 2011f] Fakhfakh, N., Verjus, H., Pourraz, F., and Moreaux, P. (2011f). Measuring the satisfaction degree of quality attributes requirements for services orchestrations. In *in The Fourth International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ 2011)*, pages 89–94, Budapest, Hungary.

[Figueira et al., 2005] Figueira, J., Greco, S., and Ehrgott, M. (2005). *Multiple criteria decision analysis : state of the art surveys.* International series in operations research & management science. Springer.

[Forman and Selly, 2001] Forman, E. H. and Selly, M. A. (2001). *Decision by objectives : how to convince others that you are right.* World Scientific Pub Co Inc.

[Gallotti et al., 2008] Gallotti, S., Ghezzi, C., Mirandola, R., and Tamburrelli, G. (2008). Quality prediction of service compositions through probabilistic model checking. In *QoSA '08 : Proceedings of the 4th International Conference on Quality of Software-Architectures*, pages 119–134, Berlin, Heidelberg. Springer-Verlag.

[Grabisch, 1997] Grabisch, M. (1997). k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets Syst.*, 92 :167–189.

[Grabisch, 2006] Grabisch, M. (2006). L'utilisation de l'intégrale de choquet en aide multi-critère à la décision. *Newsletter of the European Working Group " Multicriteria Aid for Decisions"*, 3(14) :5–10.

[Grabisch et al., 2002] Grabisch, M., Duchêne, J., Lino, F., and Perny, P. (2002). Subjective Evaluation of Discomfort in Sitting Positions. *Fuzzy Optimization and Decision Making*, pages 287–312.

[Grabisch and Labreuche, 2005] Grabisch, M. and Labreuche, C. (2005). Fuzzy measures and integrals in mcda. In *Multiple Criteria Decision Analysis : State of the Art Surveys*, volume 78 of *International Series in Operations Research &amp ; Management Science*, pages 563–604. Springer New York.

[Grabisch and Labreuche, 2008] Grabisch, M. and Labreuche, C. (2008). A decade of application of the choquet and sugeno integrals in multi-criteria decision aid. *4OR : A Quarterly Journal of Operations Research*, 6 :1–44.

[Grove, 1991] Grove, A. C. (1991). *An Introduction to the Laplace Transform and the Z Transform.* Prentice Hall.

[Haas and Brown, 2004] Haas, H. and Brown, A. (2004). W3C working group note : Web services glossary. http ://www.w3.org/TR/ws-gloss/.

[He et al., 2008] He, Q., Yan, J., Jin, H., and Yang, Y. (2008). Adaptation of web service composition based on workflow patterns. In *Proceedings of the 6th International Conference on Service-Oriented Computing*, ICSOC '08, pages 22–37, Berlin, Heidelberg. Springer-Verlag.

[Henriet, 2000] Henriet, L. (2000). *Systèmes d'évaluation et de classification multicritères pour l'aide à la décision : construction de modèles et procédures d'affectation.* PhD thesis, Université Paris Dauphine.

[Herssens et al., 2008] Herssens, C., Jureta, I. J., and Faulkner, S. (2008). Capturing and using qos relationships to improve service selection. In *Proceedings of the 20th international conference on Advanced Information Systems Engineering*, CAiSE '08, pages 312–327, Berlin, Heidelberg. Springer-Verlag.

[Hirel et al., 2000] Hirel, C., Tuffin, B., and Trivedi, K. (2000). Spnp : Stochastic petri nets. version 6.0. In Haverkort, B., Bohnenkamp, H., and Smith, C., editors, *Computer Performance Evaluation.Modelling Techniques and Tools*, volume 1786 of *Lecture Notes in Computer Science*, pages 354–357. Springer Berlin / Heidelberg.

[Hofstede et al., 2009] Hofstede, A., Aalst, W., Adams, M., and Russell, N. (2009). *Modern Business Process Automation : YAWL and Its Support Environment*. Springer.

[Huang et al., 2009] Huang, A. F. M., Lan, C.-W., and Yang, S. J. H. (2009). An optimal qos-based web service selection scheme. *Inf. Sci.*, 179 :3309–3322.

[hung Vu et al., 2005] hung Vu, L., Hauswirth, M., and Aberer, K. (2005). Qos-based service selection and ranking with trust and reputation management. In *in Proceedings of the Cooperative Information System Conference (CoopIS'05*, pages 446–483.

[Hwang et al., 2004] Hwang, S.-Y., Wang, H., Srivastava, J., and Paul, R. (2004). A probabilistic qos model and computation framework for web services-based workflows. In Atzeni, P., Chu, W., Lu, H., Zhou, S., and Ling, T.-W., editors, *Conceptual Modeling - ER 2004*, volume 3288 of *Lecture Notes in Computer Science*, pages 596–609. Springer Berlin / Heidelberg.

[Hwang et al., 2007] Hwang, S.-Y., Wang, H., Tang, J., and Srivastava, J. (2007). A probabilistic approach to modeling and estimating the qos of web-services-based workflows. *Inf. Sci.*, 177 :5484–5503.

[IEEE, 1990] IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*.

[ISA-France, 2012] ISA-France (2012). ISA-95 : MES et communication avec les ERPs.

[ISO/TC 176, 2008] ISO/TC 176 (2008). Quality management systems – requirements.

[ISO/TC 179/SC3, 2010] ISO/TC 179/SC3 (2010). Quality management – customer satisfaction – guidelines for monitoring and measuring. http ://sfkvalitet.se/files/N302ISODIS10004.pdf.

[Jaeger et al., 2004] Jaeger, M. C., Rojec-Goldmann, G., and Muhl, G. (2004). Qos aggregation for web service composition using workflow patterns. In *EDOC '04 : Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International*, pages 149–159, Washington, DC, USA. IEEE Computer Society.

[Jaeger et al., 2005] Jaeger, M. C., Rojec Goldmann, G., and Muhl, G. (2005). Qos aggregation in web service compositions. In *EEE '05 : Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service*, pages 181–185, Washington, DC, USA. IEEE Computer Society.

[Jensen, 1995] Jensen, K. (1995). *Colored Petri Nets : Basic Concepts, Analysis Methods and Practical Use*, volume 1 of *Monographs in Theoretical Computer Science a Series of Eacts*. Springer-Verlag.

[Kall and Wallace, 1994] Kall, P. and Wallace, S. (1994). *Stochastic programming*. Wiley-Interscience series in systems and optimization. Wiley.

[Karastoyanova and Buchmann, 2004] Karastoyanova, D. and Buchmann, A. (2004). Extending web service flow models to provide for adaptability. In *Proceedings of OOPSLA'04 Workshop on "Best Practices And Methodologies in Service-Oriented Architectures : Paving the way to Web-Services Sucess*.

[Kattepur et al., 2010] Kattepur, A., Sen, S., Baudry, B., Benveniste, A., and Jard, C. (2010). Variability modeling and qos analysis of web services orchestrations. In *Proceedings of the 2010 IEEE International Conference on Web Services*, ICWS '10, pages 99 –106, Washington, DC, USA. IEEE Computer Society.

[Keller and Ludwig, 2003] Keller, A. and Ludwig, H. (2003). The wsla framework : Specifying and monitoring service level agreements for web services. *J. Netw. Syst. Manage.*, 11(1) :57–81.

[Kesraoui, 2010] Kesraoui, H. (2010). Notions d'asservissement et de régulation.

[Khamès, 2010] Khamès, D. (2010). Logiciel de gestion d'atelier mes. In *Innovation & Industrie*, number 28 in Innovation & Industrie.

[Kiepuszewski et al., 2000] Kiepuszewski, B., Hofstede, A. H. M. t., and Bussler, C. (2000). On structured workflow modelling. In *Proceedings of the 12th International Conference on Advanced Information Systems Engineering*, CAiSE '00, pages 431–445, London, UK. Springer-Verlag.

[Krantz et al., 2006] Krantz, D., Suppes, P., Luce, R., and Tversky, A. (2006). *Foundations of measurement : Additive and polynomial representations*. Foundations of Measurement. Dover Publications.

[Labreuche and Grabisch, 2003] Labreuche, C. and Grabisch, M. (2003). The choquet integral for the aggregation of interval scales in multicriteria decision making. *Fuzzy Sets and Systems*, 137(1) :11 – 26. Preference Modelling and Applications.

[Lapadula et al., 2008] Lapadula, A., Rosario, P., and Tiezzi, F. (2008). A formal account of WS-BPEL. In *Proc. 10th international conference on Coordination Models and Languages (COORDINATION'08)*, volume 5052 of *Lecture Notes in Computer Science*, pages 199–215. Springer.

[Le Blevec, 2008] Le Blevec, Y. (2008). *Towards the conception of extended information systems organized around a web service platform.* PhD thesis, LIRIS, INSA de Lyon/Université Claude Bernard Lyon 1/Université de Lumière Lyon 2/École Centrale de Lyon.

[Lee et al., 2003] Lee, K., Jeon, J., Lee, W., and Park, S.-W. (2003). Qos for web services : Requirements and possible approaches. http ://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

[Leitner et al., 2009] Leitner, P., Wetzstein, B., Rosenberg, F., Michlmayr, A., Dustdar, S., and Leymann, F. (2009). Runtime prediction of service level agreement violations for composite services. In *ICSOC/ServiceWave Workshops*, pages 176–186.

[Liu et al., 2004] Liu, Y., Ngu, A. H., and Zeng, L. Z. (2004). Qos computation and policing in dynamic web service selection. In *WWW Alt. '04 : Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73, New York, NY, USA. ACM.

[Lubinski, 2008] Lubinski, T. (2008). Business activity monitoring : Process control for the enterprise. http ://www.bangalore.com/pms5.pdf.

[Ludwig and Franczyk, 2008] Ludwig, A. and Franczyk, B. (2008). Cosma — an approach for managing slas in composite services. In *Proceedings of the 6th International Conference on Service-Oriented Computing*, ICSOC '08, pages 626–632, Berlin, Heidelberg. Springer-Verlag.

[Ludwig et al., 2009] Ludwig, A., Hering, T., Kluge, R., and Franczyk, B. (2009). A case study on managing slas in composite services with cosma. In *BPSC*, pages 192–206.

[Ludwig et al., 2003] Ludwig, H., Keller, A., Dan, A., P. King, R., and Franck, R. (2003). Web service level agreements (WSLA) language specification. http ://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

[Mani and Nagarajan, 2002] Mani, A. and Nagarajan, A. (2002). Understanding quality of service for web services. http ://www.ibm.com/developerworks/library/ws-quality.html.

[Marichal, 2000] Marichal, J.-L. (2000). An axiomatic approach of the discrete choquet integral as a tool to aggregate interacting criteria. *IEEE Transactions on Fuzzy Systems*, 8(6) :800–807.

[Marichal, 2002] Marichal, J.-L. (2002). *Aggregation of interacting criteria by means of the discrete Choquet integral*, pages 224–244. Physica-Verlag GmbH, Heidelberg, Germany, Germany.

[Marichal, 99] Marichal, J.-L. (99). Agrégation de critères interactifs au moyen de l'intégrale de choquet discrète. In *Rencontres francophones sur la logique floue et ses applications (LFA'99)*, pages 130–140, Valenciennes, France.

[Mathworks, 2012] Mathworks (2012). Web deployment of MATLAB applications guide. http ://www.mathworks.fr/support/tech-notes/1600/1608.html.

[Mayag et al., 2011] Mayag, B., Grabisch, M., and Labreuche, C. (2011). A representation of preferences by the choquet integral with respect to a 2-additive capacity. *Theory and Decision*, 71 :297–324.

[Menascé, 2003] Menascé, D. A. (2003). Automatic qos control. *IEEE Internet Computing*, 7 :92–95.

[Menascé et al., 2010] Menascé, D. A., Casalicchio, E., and Dubey, V. (2010). On optimal service selection in service oriented architectures. *Performance Evaluation*, 67(8) :659 – 675. Special Issue on Software and Performance.

[Mendling et al., 2003] Mendling, J., Muller, M., and Wirtschaftsinformatik, L. (2003). A comparison of bpml and bpel4ws. In *In : Proc. 1st Conf. Berliner XML-Tage*, pages 305–316.

[Miranda et al., 2002] Miranda, P., Grabisch, M., and Gil, P. (2002). p-symmetric fuzzy measures. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10 :105–123.

[Nasri et al., 2011] Nasri, I., Habchi, G., and Boukezzoula, R. (2011). Scheduling and Control Modelling of HVLV Systems Using Max-Plus Algebra. In *5th International Workshop on Verification and Evaluation of Computer and Communication Systems*, pages 62–71, Tunis, Tunisie.

[Object Management Group, 2007] Object Management Group (2007). The uml profile for modelling and analysis of real-time and embedded systems.

[Object Management Group, 2008] Object Management Group (2008). UML profile for modelling quality of service and fault tolerance characteristics and mechanisms. http ://www.omg.org/spec/QFTP/1.1/PDF.

[Oracle, 2009] Oracle (2009). Oracle business activity monitoring.

[Papazoglou and Heuvel, 2007] Papazoglou, M. and Heuvel, W. (2007). Service oriented architectures : approaches, technologies and research issues. *The VLDB Journal, vol. 16, pp. 389-415.*

[Papazoglou et al., 2006] Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F., and Krämer, B. J. (2006). Service-oriented computing : A research roadmap. In Cubera, F., Krämer, B. J., and Papazoglou, M. P., editors, *Service Oriented Computing (SOC)*, number 05462 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany. http ://drops.dagstuhl.de/opus/volltexte/2006/524 [date of citation : 2006-01-01].

[Parasuraman et al., 1994] Parasuraman, A., Zeithaml, V. A., and Berry, L. L. (1994). Alternative scales for measuring service quality : A comparative assessment based on psychometric and diagnostic criteria. *Journal of Retailing*, 70(3) :201 – 230.

[Park, 2004] Park, K. I. (2004). *QoS in Packet Networks.* Springer.

[Pegoraro et al., 2008] Pegoraro, R., Ben Halima, R., Drira, K., Guennoun, K., and Rosario, J. M. (2008). A framework for monitoring and runtime recovery of web service-based applications. *10th International Conference on Enterprise Information Systems (ICEIS 2008).*

[Peltz, 2003] Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10) :46–52.

[Pignon and Labreuche, 2007] Pignon, J. and Labreuche, C. (2007). A methodological approach for operational and technical experimentation based evaluation of systems architectures. In *International Conference on Software & Systems Engineering and their Appllications (ICSSEA)*, pages 4–6, Paris, France.

[Pourraz, 2007] Pourraz, F. (2007). *Diapason : une approche formelle et centrée architecture pour la composition évolutive de services Web.* PhD thesis, LISTIC, Université de Savoie.

[Pourraz and Verjus, 2007] Pourraz, F. and Verjus, H. (2007). Diapason : an engineering environment for designing, enacting and evolving service-oriented architectures. In *International Conference on Software Engineering Advances (ICSEA 2007)*, pages 23–30, France. IEEE Computer Society.

[Pourraz and Verjus, 2008] Pourraz, F. and Verjus, H. (2008). *Managing Service-Based EAI Architectures Evolution Using a Formal Architecture-Centric Approach*, volume 3 of *Lecture Notes in Business Information Processing*, pages 269–280. Springer Berlin Heidelberg.

[Qiao et al., 2009] Qiao, M., Khendek, F., Serhani, A., Dssouli, R., and Glitho, R. (2009). An architecture for automatic qos adaptation for composite web services. *International Journal of Web Services Practrices*, 4(1) :18–27.

[Quynh and Thang, 2009] Quynh, P. and Thang, H. (2009). Dynamic coupling metrics for service–oriented software. *International Journal of Computer Science and Engineering*, 3(1) :46–46.

[Ran, 2003] Ran, S. (2003). A model for web services discovery with qos. *SIGecom Exch.*, 4(1) :1–10.

[Rosario et al., 2008] Rosario, S., Benveniste, A., Haar, S., and Jard, C. (2008). Probabilistic qos and soft contracts for transaction-based web services orchestrations. *IEEE Trans. Serv. Comput.*, 1 :187–200.

[Rosario et al., 2009a] Rosario, S., Benveniste, A., and Jard, C. (2009a). Flexible probabilistic qos management of transaction based web services orchestrations. In *Proceedings of the 2009 IEEE International Conference on Web Services*, ICWS '09, pages 107–114, Washington, DC, USA. IEEE Computer Society.

[Rosario et al., 2009b] Rosario, S., Benveniste, A., and Jard, C. (2009b). Monitoring probabilistic slas in web service orchestrations. In *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, IM'09, pages 474–481, Piscataway, NJ, USA. IEEE Press.

[Rosario et al., 2010] Rosario, S., Benveniste, A., and Jard, C. (2010). Flexible probabilistic qos management of orchestrations. *International Journal of Web Service Research*, 7(2) :21–42.

[Rosenberg, 2009] Rosenberg, F. (2009). *QoS-Aware Composition of Adaptive Service-Oriented Systems*. PhD thesis, Technical University Vienna, Austria.

[Rosenberg et al., 2009] Rosenberg, F., Celikovic, P., Michlmayr, A., Leitner, P., and Dustdar, S. (2009). An end-to-end approach for qos-aware service composition. In *Proceedings of the 2009 IEEE International Enterprise Distributed Object Computing Conference (edoc 2009)*, EDOC '09, pages 151–160, Washington, DC, USA. IEEE Computer Society.

[Roy, 1985] Roy, B. (1985). *Méthodologie multicritère d'aide à la décision*. Economica, Paris.

[Roy, 2009] Roy, B. (2009). àpropos de la signification des dépendances entre critères : quelle place et quels modes de prise en compte pour l'aide à la décision ? *RAIRO - Operations Research*, 43(03) :255–275.

[Russell et al., 2006a] Russell, N., Arthur, van der Aalst, W. M. P., and Mulyar, N. (2006a). Workflow control-flow patterns : A revised view. Technical report, BPMcenter.org.

[Russell et al., 2007a] Russell, N., ter Hofstede, A. H., van der Aalst, W., and Edmond, D. (2007a). newYAWL : Achieving comprehensive patterns support in workflow for the contro-flow, data and ressource perspectives. Technical report, BPM Center.

[Russell et al., 2007b] Russell, N., ter Hofstede, A. H., and van der Aalst, W. M. (2007b). newyawl : Specifying a workflow reference language using coloured petri nets. In Jensen, K., editor, *Eighth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools - CPN'07*, Aarhus, Denmark. Department of Computer Science, University of Aarhus, Denmark. The contents of this conference can be freely accessed online via the conference's web page (see hypertext link).

[Russell et al., 2006b] Russell, N., ter Hofstede, A. H. M., van der Aalst, W. M. P., and Mulyar, N. (2006b). Workflow control-flow patterns : A revised view. Technical report, BPM Center Report BPM-06-22 , BPMcenter.org.

[Sahai et al., 2002] Sahai, A., Machiraju, V., Sayal, M., Moorsel, A. P. A. v., and Casati, F. (2002). Automated sla monitoring for web services. In *Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems : Operations and Management : Management Technologies for E-Commerce and E-Business Applications*, DSOM '02, pages 28–41, London, UK. Springer-Verlag.

[Sathya et al., 2011] Sathya, M., Swarnamuqi, M., Dhavachelvan, P., and Sureshkumar, G. (2011). Evaluation of QoS based web-service selection techniques for service composition. *International Journal of Software Engineering (IJSE)*, 1(5) :73–90.

[Sato and Trivedi, 2007] Sato, N. and Trivedi, K. S. (2007). Stochastic modeling of composite web services for closed-form analysis of their performance and reliability bottlenecks. In *ICSOC '07 : Proceedings of the 5th international conference on Service-Oriented Computing*, pages 107–118, Berlin, Heidelberg. Springer-Verlag.

[Serhani et al., 2005] Serhani, M. A., Dssouli, R., Hafid, A., and Sahraoui, H. (2005). A qos broker based architecture for efficient web services selection. pages 113–120.

[Shapiro et al., 2009] Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2009). *Lectures on stochastic programming : modeling and theory*. MPS-SIAM series on optimization. Society for Industrial and Applied Mathematics.

[Srinivasan and Treadwell, 2005] Srinivasan, L. and Treadwell, J. (2005). An overview of service-oriented architecture, web services and grid computing. *HP Software Global Business Unit*.

[Systems, 2010] Systems, A. (2010). QoS implementation in WiMAX networks (ieee 802. 16-2009). http ://www.albentia.com/Docs/WP_EN/ALB-W-000001enA3-QoS.pdf.

[Szydlo and Zielinski, 2008] Szydlo, T. and Zielinski, K. (2008). Method of adaptive quality control in service oriented architectures. In *Proceedings of the 8th international conference on Computational Science, Part I*, ICCS '08, pages 307–316, Berlin, Heidelberg. Springer-Verlag.

[Taher et al., 2005a] Taher, L., Basha, R., and El Khatib, H. (2005a). A framework and qos matchmaking algorithm for dynamic web service selection. *The second International Conference on Innovations in Information Technology (IIT'05)*.

[Taher et al., 2005b] Taher, L., Basha, R., and El Khatib, H. (2005b). Qos information & computation (qos-ic) framework for qos-based discovery of web services. *UPGRADE*, 6(4).

[TC, 2007] TC, O. W. (2007). Web services business execution language version 2.0. http ://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

[Thomas, 2007] Thomas, E. (2007). *SOA Principles of Service Design (Prentice Hall Service-Oriented Computing Series from Thomas ERL)*. Prentice Hall International, 1 edition.

[Tosic et al., 2002a] Tosic, V., Pagurek, B., Esfandiari, B., Patel, K., and Ma, W. (2002a). Web service offerings language (wsol) and web service composition management (wscm). In *In Proc. of the Object- Oriented Web Services Workshop at OOPSLA 2002*.

[Tosic et al., 2002b] Tosic, V., Patel, K., and Pagurek, B. (2002b). Wsol - web service offerings language. In *Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*, CAiSE '02/ WES '02, pages 57–67, London, UK, UK. Springer-Verlag.

[van Breugel and Koshkina, 2006] van Breugel, F. and Koshkina, M. (2006). Models and verification of bpel. Technical report, York University, Toronto, M3J IP3, Canada.

[van der Aalst and Berens, 2001] van der Aalst, W. and Berens, P. (2001). Beyond workflow management : Product-driven case handling.

[van der Aalst et al., 2002] van der Aalst, W., Dumas, M., ter Hofstede, A., and Wohed, P. (2002). Pattern-based analysis of bpml (and wsci). Technical report.

[van der Aalst and Hofstede, 2002] van der Aalst, W. and Hofstede, A. H. M. T. (2002). Workflow patterns : On the expressive power of (petri-net-based) workflow languages. In *of DAIMI, University of Aarhus*, pages 1–20.

[van der Aalst and ter Hofstede, 1999] van der Aalst, W. and ter Hofstede, A. (1999). Workflow patterns. http ://www.workflowpatterns.com/.

[van der Aalst and ter Hofstede, 2005] van der Aalst, W. and ter Hofstede, A. (2005). Yawl : Yet another workflow language. *Information Systems*.

[van der Aalst et al., 2003] van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., and Barros, A. (2003). Workflow patterns. In *Distributed and Parallel Databases*, volume 14 (1), pages 5–51.

[Vasko and Dustdar, 2004] Vasko, M. and Dustdar, S. (2004). An analysis of web services workflow patterns in collaxa. In *in Collaxa. European Conference on Web services (ECOWS)*. Springer LNCS.

[Verjus, 2011] Verjus, H. (2011). Mestria : une solution mes agile à base de services. *JITEC (Journal d'Information TEChnologique)*, July-August(151) :4.

[Verjus and Pourraz, 2008] Verjus, H. and Pourraz, F. (2008). Diapason : A formal approach for supporting agile and evolvable information system service-based architectures. In *Proceedings of the 10th International Conference on Enterprise Information Systems (ICEIS'08)*, volume ISAS-2, pages 76–81, Barcelona, Spain. INSTICC-WfMC-AAAI.

[Verjus et al., 2011] Verjus, H., Pourraz, F., and Fakhfakh, N. (2011). Cadre conceptuel pour la modélisation et la supervision d'architectures à base de services. Une approche pour l'ingénierie des SI à base de services. *Développement des SI à base de modèles : exigences, traçabilité et co-conception - Ingénierie des Systèmes d'Information - Revue des sciences et technologies de l'information*, 16(5) :43–72.

[Wang et al., 2009] Wang, Q., Shao, J., Deng, F., Liu, Y., Li, M., Han, J., and Mei, H. (2009). An online monitoring approach for web service requirements. *IEEE Trans. Serv. Comput.*, 2 :338–351.

[WBF, 2012] WBF, T. O. f. P. T. (2012). Business to manufacturing markup language (B2MML). http ://wbforg.affiniscape.com/displaycommon.cfm ?an=1&subarticlenbr=99.

[Wetzstein et al., 2008] Wetzstein, B., Karastoyanova, D., and Leymann, F. (2008). Towards Management of SLA-Aware Business Processes Based on Key Performance Indicators. In *9th Workshop on Business Process Modeling, Development and Support (BPMDS'08) - Business Process Life-Cycle :Design, Deployment, Operation & Evaluation.*

[WfMC, 1999] WfMC (1999). Workflow management coalition : Terminology and glossary.

[WfMC, 2011] WfMC (2011). Workflow Management Coalition. http ://www.wfmc.org/.

[Zeithaml et al., 1993] Zeithaml, V., Berry, L., and Parasuraman, A. (1993). The nature and determinants of customer expectations of service. *Journal of the Academy of Marketing Science*, 21 :1–12. 10.1177/0092070393211001.

[Zeng et al., 2003] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q. Z. (2003). Quality driven web services composition. pages 411–421.

[Zeng et al., 2004] Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30 :311–327.

[Zhang et al., 2008] Zhang, P., Li, B., Muccini, H., and Sun, M. (2008). Advanced web and networktechnologies, and applications. chapter An Approach to Monitor Scenario-Based Temporal Properties in Web Service Compositions, pages 144–154. Springer-Verlag, Berlin, Heidelberg.

[Zhenehen et al., 2011] Zhenehen, Z., Lage, N., and Crespi, N. (2011). User-centric service selection, integration and management through daily events. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 94–99.

[Zheng and Lyu, 2010] Zheng, Z. and Lyu, M. R. (2010). An adaptive qos-aware fault tolerance strategy for web services. *Empirical Software Engineering*, 15 :323–345.

[Zhong and Qi, 2006] Zhong, D. and Qi, Z. (2006). A petri net based approach for reliability prediction of web services. In *OTM Workshops (1)*, pages 116–125.

[Zo et al., 2007] Zo, H., Nazareth, D. L., and Jain, H. K. (2007). Measuring reliability of applications composed of web services. In *HICSS '07 : Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, page 278c, Washington, DC, USA. IEEE Computer Society.

Résumé : La qualité de service est devenue aujourd'hui une notion incontournable dans le développement des applications logicielles, en particulier dans le cadre des architectures orientées services. Les travaux de cette thèse se focalisent sur la supervision de la qualité de service des applications orientées services, définies sous forme d'orchestrations de services. L'approche de supervision proposée dans ce contexte est générique. Elle repose sur des patrons de flux de contrôle des orchestrations de services pouvant être implémentés en intégralité ou en partie par tout langage d'orchestration de services. D'autre part, elle ne pose aucune restriction, ni sur les attributs qualité à surveiller par le système de supervision, ni sur leurs représentations. Cette approche de supervision se distingue des approches existantes par l'exploitation d'un modèle de préférences orienté utilisateur, permettant de représenter fidèlement la satisfaction de ce dernier. Le degré de satisfaction, issu du modèle de préférences, constitue une information de haut niveau représentant la qualité globale de l'orchestration étudiée. Sur la base de ce degré de satisfaction, de nouvelles stratégies de surveillance sont proposées afin de satisfaire les attentes de l'utilisateur. L'élaboration du modèle de préférences exploite la méthode d'aide à la décision multi-critères MACBETH étendue avec l'opérateur d'agrégation de l'intégrale de Choquet 2-additive.

Une illustration de l'approche de supervision a été réalisée sur une orchestration de services, représentant un processus industriel dans le domaine du pilotage d'atelier de production. Les travaux de cette thèse ont été réalisés dans le cadre d'un projet *R&D* regroupant sept éditeurs de progiciels dans le domaine du MES (*Manufacturing Execution System*).

Mots-clefs : supervision, orchestrations de services, degré de satisfaction, qualité de service, règles d'agrégation de patrons de *workflow*, méthode d'aide à la décision multicritère, architectures orientées services.

Abstract: Quality of Service (QoS) is an important issue today in the developement of software applications, especially in the context of Service-Oriented Architectures (SOA).

The work of this thesis focuses on QoS supervision of service-oriented applications, defined as service orchestrations. The proposed supervision approach is generic. It is based on workflow control-flow patterns, which can be entirety or partillay implemented by any service orchestration language. On the other hand, it does not make any restrictions, neither on the monitored QoS attributes , nor on their representations. This supervision approach is based on a user-oriented preferences model, that represents faithfully the user satisfaction. The satisfaction degree derived from the preferences model is a high-level information representing the overall quality of the orchestration. New monitoring strategies are proposed on the basis of this satisfaction degree in order to satisfy user expectations. The elaboration of the preferences model uses the MACBETH method extended to the 2-additive Choquet integral operator as a multi criteria decision aiding method.

An illustration of the approach is carried out on a service orchestration representing a Manufacturing Execution System (MES) process. This work was realized in a *R&D* project involving seven software vendors in the field of MES.

Keywords: Supervision, Service Orchestration, Satisfaction Degree, Quality of Service, Workflow Pattern Aggregation Rules, Multicriteria Decision Aiding Method, Service-Oriented Architectures.